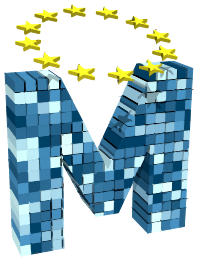


Project title: Multi-Owner data Sharing for Analytics and Integration respecting Confidentiality and OWNer control
Project acronym: MOSAICrOWN
Funding scheme: H2020-ICT-2018-2
Topic: ICT-13-2018-2019
Project duration: January 2019 – December 2021

D2.4

Use Case Prototypes

Editors: Jonas Böhler (SAP SE)
Reviewers: Stefano Paraboschi (UNIBG)
Pierangela Samarati (UNIMI)

Abstract

This deliverable details the final tools and prototype application for each MOSAICrOWN use case. For each use case, the application scenario is presented, required background information is described, and the tools and applications to fulfill the use case are detailed. Use Case 1, by EISI, supports data collection and sharing in the context of autonomous vehicles. Use Case 2, by MC, focuses on data analysis for financial data markets. Use Case 3, by SAP SE, supports data sharing and statistical analysis in the context of cloud-based data markets.

Type	Identifier	Dissemination	Date
Deliverable	D2.4	Public	2021.12.31

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825333.

MOSAICrOWN Consortium

1.	Università degli Studi di Milano	UNIMI	Italy
2.	EMC Information Systems International	EISI	Ireland
3.	Mastercard Europe	MC	Belgium
4.	SAP SE	SAP SE	Germany
5.	Università degli Studi di Bergamo	UNIBG	Italy
6.	GEIE ERCIM (Host of the W3C)	W3C	France

Versions

Version	Date	Description
0.1	2021.11.29	Initial Release
0.2	2021.12.20	Second Release
1.0	2021.12.31	Final Release

List of Contributors

This document contains contributions from different MOSAICrOWN partners. Contributors for the chapters of this deliverable are presented in the following table.

Chapter	Author(s)
Executive Summary	Jonas Böhler (SAP SE)
Chapter 1: Use Case 1 (EISI): Tools for ICV platforms	Aidan O Mahony (EISI), Alan Barnett (EISI), Merry Globin (EISI)
Chapter 2: Use Case 2 (MC): Tools for financial data markets	Flora Giusto (MC), Saverio Mucci (MC)
Chapter 3: Use Case 3 (SAP SE): Privacy-preserving tools for cloud-based data markets	Jonas Böhler (SAP SE)
Chapter 4: Conclusions	Jonas Böhler (SAP SE)

Contents

Executive Summary **11**

1 Use Case 1 (EISI): Tools for ICV platforms **13**
1.1 Introduction . . . 13
1.2 Use Case Background . . . 14
1.3 Automotive Tools . . . 15
1.3.1 Android Auto . . . 16
1.3.2 MOSAICrOWN Governance Framework Modifications . . . 18
1.3.3 Monetization of Private Data via Automotive Tools . . . 19
1.4 Web UI . . . 19
1.4.1 Pug - A Template Engine . . . 19
1.4.2 Web UI Structure . . . 21
1.4.3 Maps . . . 22
1.4.4 Data Access Types . . . 22
1.5 Policy Engine . . . 26
1.5.1 Overview of Tools . . . 26
1.5.2 Modification of Tools . . . 26
1.6 Encrypted File System . . . 27
1.6.1 Overview of Tools . . . 28
1.6.2 Modification of Tools . . . 28
1.7 Summary . . . 30

2 Use Case 2 (MC): Tools for financial data markets **31**
2.1 Background . . . 32
2.1.1 Overview . . . 32
2.2 Solutions . . . 33
2.3 Architecture and Components . . . 33
2.3.1 Client-Server Model . . . 33
2.3.2 User Interface . . . 35
2.3.3 Application Review . . . 39
2.3.4 Process Diagram . . . 40
2.3.5 Sequence Process . . . 40
2.4 Summary . . . 40

3 Use Case 3 (SAP SE): Privacy-preserving tools for cloud-based data markets 43
3.1 Introduction . . . 43
3.2 Background . . . 44
3.2.1 Background on Differential Privacy . . . 44
3.2.2 Background on Secure Computation . . . 46
3.3 Solutions . . . 48
3.3.1 DPtool: Toolbox for Differential Privacy Mechanisms . . . 48
3.3.2 MIA: Analysis for Membership Inference Attack . . . 52
3.3.3 DPsc: Secure Computation of Differentially Private Statistics . . . 54
3.4 Summary . . . 57

4 Conclusions 59

Bibliography 60

List of Figures

1.1 Overview of UC1's data flow . . . 13
1.2 Android Auto installation in Polestar 2 Electric Vehicle [Pol] . . . 17
1.3 Android Auto interface at the start of the journey. The driver has the option to select from any of the three policies . . . 20
1.4 Android Auto interface after selecting policy for electricity cost per KWh. The driver has the opportunity the pay less for electric charge depending on how much PII they are allowing to collect . . . 21
1.5 A basic HERE Map . . . 22
1.6 Dashboard for MOSAICrOWN Cloud Provider . . . 23
1.7 Dashboard for vehicle driver . . . 24
1.8 Dashboard for Fleet Owner . . . 25
1.9 Dashboard for EV Infrastructure Provider . . . 25
1.10 Policy Engine API data flow . . . 27
1.11 Parameters of a subjectless API endpoint query . . . 27
1.12 Architecture of initial FreyaFS utility . . . 28
1.13 Architecture of modified FreyaFS utility . . . 29

2.1 Configuration file example . . . 34
2.2 Three tier architecture . . . 34
2.3 Select privacy policy and user journey . . . 36
2.4 Error message and data upload structure template . . . 36
2.5 Data semantic type detection . . . 37
2.6 Data wrapping techniques . . . 37
2.7 Download anonymized dataset . . . 37
2.8 Analytics section - Summary tab . . . 38
2.9 Analytics section - Overall view (1/2) . . . 38
2.10 Analytics section - Overall view (2/2) . . . 39
2.11 Analytics section - By product . . . 39
2.12 Process diagram . . . 41
2.13 Sequence process - Anonymization . . . 41
2.14 Sequence process - Analytics . . . 42

3.1 UC3 Overview . . . 43
3.2 DPtool: Step 1 – data selection and ingestion . . . 50
3.3 DPtool: Step 2 – method selection . . . 51
3.4 DPtool: Step 3 – method parameterization . . . 51
3.5 DPtool: Parameter validation and tooltip . . . 51

3.6 MIA: Step 1 – model selection 52
3.7 MIA: Step 2 – data selection 53
3.8 MIA: Step 3 – parameterization 53
3.9 Sorted data set *D* with ranks per unique datum 54

List of Tables

3.1 DPtool: Main anonymization methods 49

Executive Summary

This deliverable provides an overview of the tools and applications developed in the MOSAICrOWN project to satisfy the use cases by the industry partners (EISI, SAP SE, MC). The tools leverage the technologies developed in Work Packages 3–5. The use cases and their respective tools are summarized in the following.

Use Case 1: Tools for ICV platforms (EISI). Use Case 1 considers data protection in Intelligent Connected Vehicle (ICV) platforms to facilitate data exchange between a fleet manager, Electric Vehicles (EV), and their charging infrastructure to gather insights and, e.g., inform infrastructure planning and allocation for EV charging.

The tools for Use Case 1 provide a platform for storing and accessing data securely and are summarized next. The *automotive tools* facilitate data ingestion from the electric vehicle to the data market and cover the non-functional requirements of the data economy. The tools include an application which can be integrated into the electric vehicle. The *web tools* facilitate the data access and authorization mechanisms depending on the access rights of each user via a web-based user interface. The *policy engine* parses the MOSAICrOWN policy and permits or denies requests enabling access control in the context of the ICV platform. The *encrypted filesystem* provides encryption for the secure storage of privacy related personal data. It includes tools developed by academic partners, FreyaFS.

Use Case 2: Tools for financial data markets (MC). Use Case 2 is concerned with transaction-level financial data and data wrapping techniques for the final purpose of analytics and the extraction of business insights from the data itself.

The developed application allows users, once connected to the platform via a web browser, to upload the dataset to be anonymized. The platform proposes wrapping techniques for each field (according to the policy regulation selected), once confirmed or overwritten by the user, the platform anonymizes the dataset. In more detail, the *identity component* authenticates users to use the offered services. The *semantics/wrapping component* performs a semantic analysis to detect, e.g., data types and proposes protection techniques. Users can then define and apply the wrapping techniques for each field. The *analytics component* gives users access to a dedicated dashboard to generate insights based on analytics over the anonymized data. The analysis provides aggregated results, so no granular information is presented.

Use Case 3: Privacy-preserving tools for a cloud-based data market (SAP SE). Use Case 3 operates in a cloud-based data market where businesses acting as data providers aim to perform analytics over their data while protecting the underlying information. For example, a producer and retailer want to learn supply chain insights and improve allocation of a marketing

budgets from operational data (indicators for industry benchmarking, e.g., return rates, process times) and customer experience data (ratings, purchase history) while ensuring their sensitive business and customer data is protected.

To provide privacy-preserving analytics in the context of the use case as well as appropriate guided parameterization based on sanitization techniques satisfying differential privacy, three main tools are provided: DPtool, MIA, DPsc. DPtool provides a REST API as well as graphical web-interface for the selection, parameterization and application of various differential privacy mechanisms (e.g., perturbation via Laplace or Geometric noise) on data in cvs format. MIA helps data scientists with the parameterization of differential privacy for machine learning applications as well as its evaluation via membership inference risks via a graphical web-interface. DPsc additionally leverages cryptographic techniques to only share anonymized analytics in the form of rank-based statistics (i.e., percentiles) securely computed over distributed data of multiple parties.

1. Use Case 1 (EISI): Tools for ICV platforms

This section illustrates the tools for Use Case 1 (UC1) used in the development of prototype applications. The task that this deliverable is based on is T2.3 “Testbed platform and deployment to Use Cases”, with specific reference to UC1. T2.3 leverages the technologies developed in WPs 3-5. The tools presented in this deliverable provide an automotive interface to the platform, a web-based front end to the platform, and an encrypted file system used for the secure storage of personal data.

1.1 Introduction

The emphasis of UC1, data protection in ICV (Internet Connected Vehicle) platforms, is on data ingestion, storage and analytics, taking into account data governance policy, data wrapping and sanitization.

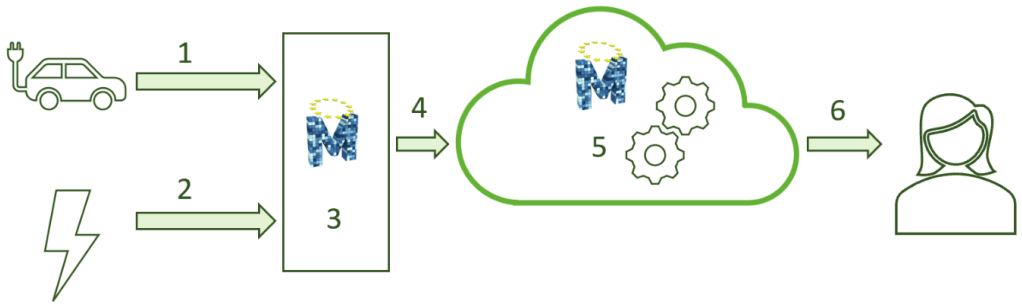


Figure 1.1: Overview of UC1’s data flow

Figure 1.1 gives an overview of UC1’s data flow, showing the exchange of data between the main components of the platform. Each stage of the data flow is described below:

1. Ingestion of ICV data into the MOSAICrOWN platform, which is obtained from the vehicles being monitored. The data is in JSON format, and is a combination of dynamic metrics and the metadata that describe them.

2. Ingestion of EV charging station data into MOSAICrOWN platform. The data is in CSV format.

3. Initial processing of data into the MOSAICrOWN platform. The metadata is extracted from the data and converted into a canonical JSON-LD (JSON Linked Data) format. As part of the process, data is cross referenced to the metadata.

4. The data and metadata are stored in the MOSAICrOWN platform. The data is stored on a high capacity file system and the metadata is stored in a knowledge base, which is implemented as an RDF triple store.

5. Analytics is performed by the MOSAICrOWN platform.

6. The data and analytics are made available to the user. The type of data and any redactions are dependent on the user's role, and consequent access rights.

The remainder of this chapter is organized as follows. Section 1.2 gives some background to UC1. Section 1.3 discusses the automotive interfaces evaluated for the platform, and how the selected interface was implemented. Section 1.4 explains the Web UI implemented by the platform, and the ways different users can interact with it. Section 1.5 covers the implementation of the MOSAICrOWN policy engine and the enhancements implemented for UC1. Section 1.6 discusses the encrypted file system for the secure storage of privacy-related personal data. Section 1.7 summarizes the outcomes of UC1.

1.2 Use Case Background

UC1 demonstrates how an automotive data market can be implemented using the MOSAICrOWN platform. The use case describes how automotive user data from electric vehicles can be safely sanitized/wrapped and monetized through the data market. The driver/data owner may participate in the data market by selecting policies to be applied to their data in exchange for discounted electric vehicle charging rates. The Web UI of the platform show cases how the privacy of data and the access rights of users can be enforced. Several critical stages in the data flow were considered when implementing the use case, including:

- Raw data ingestion
- Intermediate storage of data prior to processing
- Dynamic application of privacy-preserving techniques
- Access control and enforcement through a data market service.

One innovative feature of UC1 is the use of containers to encapsulate significant stages in the data flow. Containerizing these stages allows technologies to be swapped in and out of the solution without significantly effecting other stages, thus reducing dependencies and increasing flexibility in technology selection. Additionally, containerization greatly simplifies deployment of the solution in different environments. UC1 used Docker to implement the containerization, but any suitable containerization technology could have been used. The use case also demonstrated how the data storage could be externalized from the containers and abstracted behind an open API such as S3.

Ingestion and Filtering

The initial ingestion of vehicle, charging station and policy data is handled in Apache Nifi, an open source framework for data processing and distribution. Nifi is responsible for many of the key intermediary processes applied to the raw data, including the data/metadata extraction, the conversion of metadata and policies to RDF, and the placement of data/metadata in relevant location for access from the data market.

Nifi also incorporates the data market filter functionality whereby an appropriate privacy-preserving mechanism is first carried out prior to the data reaching the data market. This is implemented as a policy-dependent routing mechanism. Depending on the current metadata policy at ingestion time, data can be routed to a service to provide wrapping or instantiation.

Data Lake/Storage

The data lake and data market components are built on open-source solutions. Apache Hadoop is responsible for the storage of raw and treated data while Apache Jena is used to store metadata in RDF format. Data can be accessed through SPARQL queries to Jena. Access control and policy enforcement are applied before retrieving and returning data from HDFS through a WebHDFS interface.

In the following sections the tooling developed and integrated with these core platform components are detailed.

1.3 Automotive Tools

The tools described in this section address a number of requirements presented in Deliverable D2.1 "Requirements from the Use Cases". Specifically, the automotive tools facilitate data ingestion from the Electric Vehicle (EV) to the data market as well as enable the data economy non-functional requirements. Briefly, the tools take the form of an application to be integrated into the EV as well as modifications to the MOSAICrOWN governance framework. In the following, we present each component of the tools and we describe a scenario which facilitates the monetization of private data for the benefit of the data owner.

Electric Vehicle Requirements

In UC1, the connected vehicle fleet manager and the EV charging infrastructure provider want to exchange data such that they can derive mutually beneficial insights into the status of the EV charging infrastructure. To that end, the data of the EV and the behavioral data of the driver are required. The most relevant requirements are that ingestion mechanism should support real-time stream data handling (REQ-UC1-DI2) as well as ingestion from multiple concurrent sources (REQ-UC1-DI10) and that the platform should support a licensed model (REQ-UC1-DE1).

To facilitate the ingestion of data from the EV, we carried out an analysis of the available options to integrate tools into the EV, developed an application to integrate with the EV, modified the MOSAICrOWN data governance framework to ingest that data, and finally, augmented our design such that monetization opportunities are demonstrated.

In-vehicle Infotainment software

To select the tools used by UC1, we first performed a review of the existing automotive tools for driver UIs, called In-vehicle Infotainment (IVI). We evaluated the following IVI systems [CM16, Koo21], on which we comment next.

Considered IVI systems

Apple CarPlay: it is an Apple standard that enables a car radio or head unit to be a display and a controller for an iOS device.

Android Auto: it is a mobile app developed by Google to mirror features of an Android device, such as a smartphone, on a car's dashboard information and entertainment head unit.

GENIVI: it is an industry alliance involving OEMs, such as Bosch and Hyundai, working on an open-source infotainment platform for vehicles.

Tizen IVI: it is an open-source project, part of the Linux Foundation and carried by Intel, providing a free environment based on HTML5.

QNX Car platform: it allows the development of customizable apps for radio, weather and location-based systems, and web applications based on HTML5, CCS3, and JavaScript.

Windows Embedded Automotive: it allows the leveraging of Silverlight for the creation of advanced HMI interfaces.

Automotive Grade Linux: it is an open-source stack for in-vehicle infotainment based on Tizen IVI (of which it is a bootable distribution).

Evaluation of IVI systems

We carried out an evaluation of the IVI systems introduced in the previous section with a goal of identifying the most suitable IVI system to use as a prototype tool for use within an EV. This tool needed to satisfy the requirements presented in Section 1.3 and also, it should be user friendly. Our evaluation is as follows:

Apple CarPlay: Closed ecosystem, with difficulties in running development environment on Microsoft Windows based computers.

Android Auto: Depends on proprietary tools, requires Java or Kotlin knowledge.

GENIVI: Steep learning curve, experienced difficulties in installing development environment.

Tizen IVI: Steep learning curve, experienced difficulties in installing development environment.

QNX Car platform: Licensed platform, well-regarded Real-Time Operating System (RTOS).

Windows Embedded Automotive: No longer supported.

Automotive Grade Linux: Difficult to install, development of applications was too cumbersome.

Based on our evaluation of the available IVI systems, Android Auto was selected to develop the MOSAICrOWN mobile application.

1.3.1 Android Auto

Android Auto is a mobile application developed by Google to replicate the functionality of an Android device, such as a smartphone, on a car's dashboard information and entertainment head unit. It provides a driver-optimized app experience for users with an Android phone and the Android Auto app, but who do not have a vehicle that uses Android Automotive OS. As of April 2021, Android Auto is available in 42 countries. An example of an installation of Android Auto in a Polestar 2 EV is show in Figure 1.2.

Figure 1.2: Android Auto installation in Polestar 2 Electric Vehicle [Pol]

Android Software Development Kit (SDK)

Applications on Android platforms are typically established in Java programming language using the Android SDK as well as other development environments and the Android Debug Bridge (ADB). The Android SDK consists of a comprehensive set of development tools such as

- debugger
- libraries
- handset emulator
- documentation
- sample code
- tutorials

Android applications come in .apk format and saved under /data/app folder on the Android OS (this folder can only be accessed by the root user for security purposes). APK package contains .dex files (this is a set of compiled byte code files called Dalvik executables), resource files, etc. The Android Debug Bridge (ADB) is a toolkit encompassed in the Android SDK package. The ADB comprises of both client and server-side programs that communicate with one another and characteristically read through the command-line interface. However, multiple graphical user interfaces exist to control ADB.

Android Auto emulator

The Desktop Head Unit (DHU) enables the development machine to emulate an Android Auto head unit, so that a developer can run and test Android Auto apps. The DHU runs on Windows, MacOS, and Linux systems.

1.3.2 MOSAICrOWN Governance Framework Modifications

We linked the policy language introduced in Deliverable D3.3 "First Version of Policy Specification Language and Model" with data being ingested from an EV. For UC1, we decided to allow the EV driver to decide which policy is applied to their data. To simplify the interaction for the EV driver we predefined three levels of policies. These are: *a)* a policy with the most private settings described, *b)* a policy with "moderate" privacy settings stipulated, and *c)* a policy with the most permissive privacy settings described. For illustration, we present the most permissive privacy policy in Listing 1.1. Most permissive privacy policy is the policy where all the data is accessible without any restrictions to the user, regardless of their assigned role on the platform. In the Listing 1.1, we can see the permissions given to fleet manager.

```
{
  "@context": [
    "http://www.w3.org/ns/odrl.jsonld",
    "http://localhost:8000/ns/mosaicrown/namespace.jsonld"
  ],
  "@type": "Set",
  "uid": "http://dellemc.com/policy/MostPermissivePrivacyPolicy",
  "permission": [
   {
    "uid": "http://dellemc.com/policy/MostPermissivePrivacyPolicies_perm",
    "assignee": "http://dellemc.com/user/fleetmanager",
    "target": ["http://dellemc.com/icv/licensePlate",
               "http://dellemc.com/icv/vin",
               "http://dellemc.com/icv/name",
               "http://dellemc.com/icv/category",
               "http://dellemc.com/icv/type",
               "http://dellemc.com/icv/modelName",
               "http://dellemc.com/icv/modelYear",
               "http://dellemc.com/icv/colourName",
               "http://dellemc.com/icv/numberOfDoors",
               "http://dellemc.com/icv/numberOfSeats",
               "http://dellemc.com/icv/gearbox",
               "http://dellemc.com/icv/displayUnit",
               "http://dellemc.com/icv/driverSeatLocation",
               "http://dellemc.com/icv/charging/estimatedRange",
               "http://dellemc.com/icv/charging/batteryLevel",
               "http://dellemc.com/icv/diagnostic/mileage",
               "http://dellemc.com/icv/diagnostic/batteryVoltage",
               "http://dellemc.com/icv/diagnostic/speed",
               "http://dellemc.com/icv/ignition/status",
               "http://dellemc.com/icv/usage/averageFuelConsumption",
               "http://dellemc.com/icv/usage/averageWeeklyDistance",
               "http://dellemc.com/icv/usage/lastTripEnergyConsumption",
               "http://dellemc.com/icv/usage/lastTripFuelConsumption",
               "http://dellemc.com/icv/naviDestination/coordinates/
                  latitude",
               "http://dellemc.com/icv/naviDestination/coordinates/
                  longitude",
               "http://dellemc.com/icv/naviDestination/destinationName",
               "http://dellemc.com/icv/naviDestination/vehicleLocation/
                  latitude",
               "http://dellemc.com/icv/naviDestination/vehicleLocation/
                  longitude",
```

```
                "http://dellemc.com/icv/naviDestination/vehicleLocation/
                    heading",
                "http://dellemc.com/icv/naviDestination/coordinates/
                    longitude"
            ],
      "action": ["odrl:read","odrl:use","odrl:write","odrl:sell","odrl:
          sellReport"],
      "purpose": ["statistical", "marketing"]
    }
  ]
}
```

Listing 1.1: An example of a most permissive privacy policy expressed in JSON-LD

1.3.3 Monetization of Private Data via Automotive Tools

The Android application developed for UC1 allows the driver to select what policy they would like applied to their PII based on the use the data would be put to, e.g. statistical. Figure 1.3 shows an instance of the application prior to the driver charging their vehicle. The 'Incognito' option is the most private policy setting, 'Confidential' is moderate privacy setting, and 'Public' is the most permissive policy setting. Figure 1.4 shows the application when the driver presents the vehicle for charge. The driver is presented with options as to which policy the want applied to their data from that time onwards in exchange for reduced electricity cost for charging their vehicle. This policy can be applied for either a set period of time or until the next charge.

1.4 Web UI

This section explains the Web UI, implementing some of the requirements presented in the Deliverable D2.1 "Requirements from the Use Cases". The main functionality of the Web UI is to enable data access by different users (MOSAICrOWN Cloud Provider, Car Driver, Fleet Owner and EV Charging Infrastructure Provider) according to their access rights specified in the MOSAICrOWN policies. In more detail, the Web UI addresses the requirement for access control to the data with specific levels of granularity based on the policy constraints (REQ-UC1-AC1), restricting data sets and/or users for certain operations (e.g., only allows sharing; REQ-UC1-AC4), allow data sharing with specific or multiple data consumers (REQ-UC1-AC5, REQ-UC1-AC6). Furthermore, data should be accessible to all authorized consumers of the data at the same time, regardless of preferred format (REQ-UC1-DM2), and data integrity should be maintained separately in isolation (REQ-UC1-DM3).

1.4.1 Pug - A Template Engine

The Web UI is implemented using Node.js and the Pug template engine. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes server side JavaScript code [Fou]. Pug is a template engine for Node.js, which is used to render HTML pages. It is implemented in JavaScript and released under the MIT license. The Pug template engine converts the Pug code to HTML code at compile time. The benefit of using a templating engine is that it allows for the reuse of HTML page elements, while defining dynamic elements based on the data [sit]. An example of Pug code is shown below.

Figure 1.3: Android Auto interface at the start of the journey. The driver has the option to select from any of the three policies

```
doctype html
head
    title Pug
    script(type='text/javascript').
        if (foo) bar(1 + 5)
    script(src='/javascripts/jquery.js')
h1 Pug - node template engine
#container.col
    p Pug is a terse and simple templating language.
```

The above Pug code will be converted into the HTML code, by the Pug template engine.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Pug</title>
        <script type="text/javascript"> if (foo) bar(1 + 5) </script>
<script src="/javascripts/jquery.js"></script>    </head>
```

Figure 1.4: Android Auto interface after selecting policy for electricity cost per KWh. The driver has the opportunity the pay less for electric charge depending on how much PII they are allowing to collect

```
        <body>
            <h1>Pug - node template engine</h1>
            <div id="container" class="col">
                <p>Pug is a terse and simple templating language.</p>
            </div>
        </body>
    </html>
```

1.4.2 Web UI Structure

The Web UI is structured to have different views depending on which user is logged in. Users for Web UI are MOSAICrOWN Cloud Provider, Car Driver, Fleet Owner and EV Charging Infrastructure Provider. According to the access rights, the data accessible to users will vary, and to accommodate that, the Web UI presents different views. The following sections explain the different views and data that are accessible by each user.

1.4.3 Maps

A map is used to display EV charging stations and the current location of vehicles on the Web UI. The map is implemented using the HERE Maps software package, which is developed by HERE Technologies. In the Web UI, we have adapted HERE Maps to implement the specific functionality required by UC1. The data shown in the map varies according to the user logged in, and dynamically shows the vehicles tracking across the map. Section 1.4.4 details the adaptions made to HERE Maps and illustrates the different display formats base on the user's role defined on the UC1 MOSAICrOWN platform. Figure 1.5 shows an example image of a map from HERE Technologies [Tec].

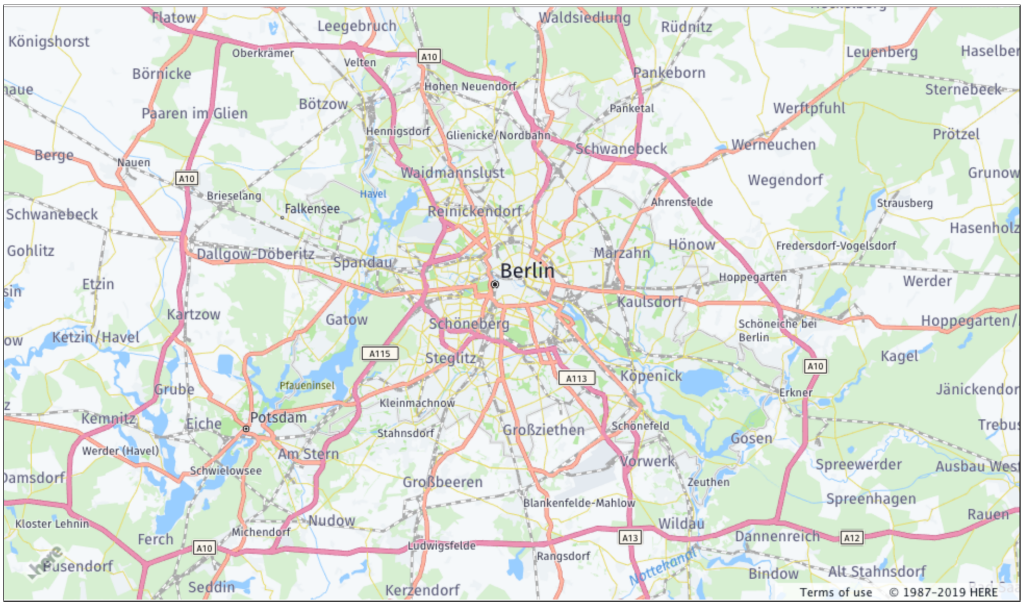


Figure 1.5: A basic HERE Map

1.4.4 Data Access Types

The data can be accessed through the Web UI using two different message formats: JSON or RDF (Resource Description Framework). The required data can be retrieved in RDF format using SPARQL queries.

SPARQL Queries

SPARQL is the standard query language for RDF triple stores, and structurally and syntactically it looks similar to SQL, but they differ semantically in their implementation. SQL searches sets of tables using a statically defined schema, whereas SPARQL searches a graph of nodes and edges by matching a set of patterns. RDF triple stores and their associated query language are often more flexible and expressive than conventional relational databases, as they are not bound by the limitations of a static schema. As the name triple store implies, RDF is defined in terms of a binary predicate, called a triple, consisting of a subject, an object, and a predicate that links them together. RDF triples are normally ordered: subject, predicate, object. The WHERE clause then consists of one or more triple patterns that the query must match, the '?' suffix indicating wildcards that can match any value. The results are returned as a list of triples that match the query.

In the listing below the SPARQL query returns any triples that have a model year equal to ‘2010’. Since the metadata knowledge base contains information about vehicles, the query will return all vehicles made in ‘2010’. All triples have a unique subject identifier, which in this case is the VIN (Vehicle Identification Number) of the vehicles. So, in this case, the query returns all vehicle VINs for vehicles made in ‘2010’. This is a very simple query provided as an illustration, but far more complex queries can be constructed, allowing the Web UI to extract very detailed and specific information.

```
SELECT ?subject ?predicate ?object WHERE {?subject <https://uri.etsi.org/
    ngsi-ld/default-context/modelYear>  "2010" .}
```

Dashboards in Web UI

Web UI has four separate views, one for each user role. Depending on the user logged in, the dashboard will display a different view and thus give a different perspective on the data. The following paragraphs present details about each user role’s dashboard.

MOSAICrOWN Cloud Provider Dashboard. The MOSAICrOWN Cloud Provider view has five tabs: Dashboard, History, Reports, SPARQL and Add New User. Figure 1.6 shows the Dashboard view. The user can see the locations and details of the EV charging stations, and the vehicle details in the dashboard. The user can also add new users in the Add New User tab, and submit custom queries for data from metadata knowledge base in the SPARQL tab.

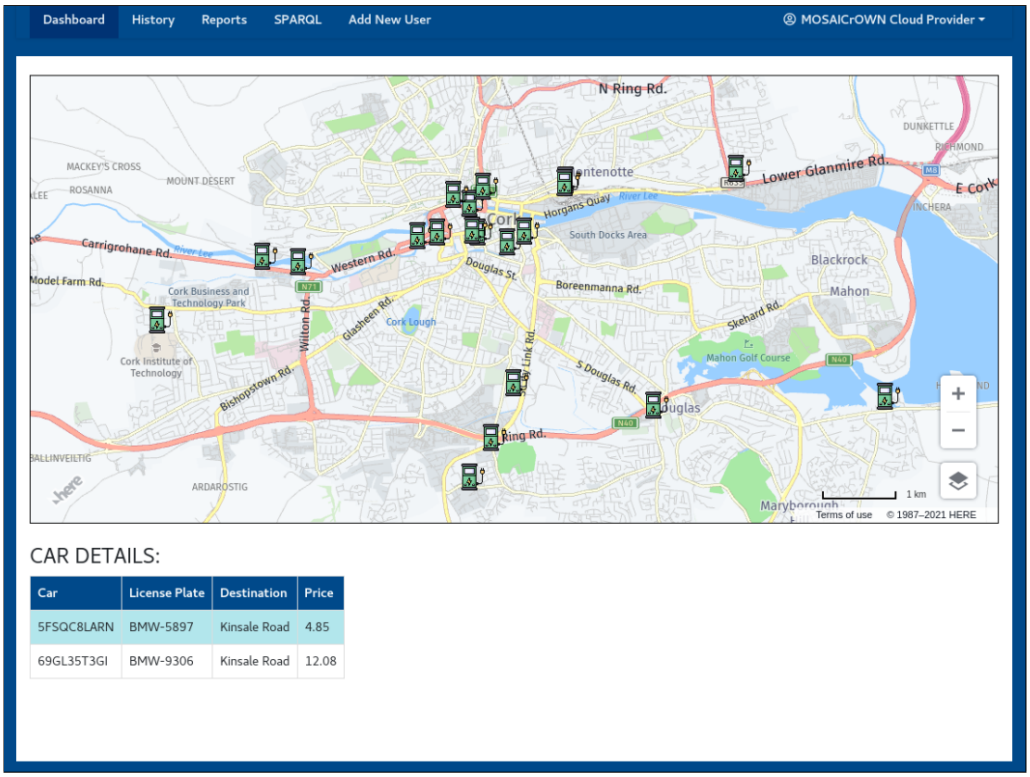


Figure 1.6: Dashboard for MOSAICrOWN Cloud Provider

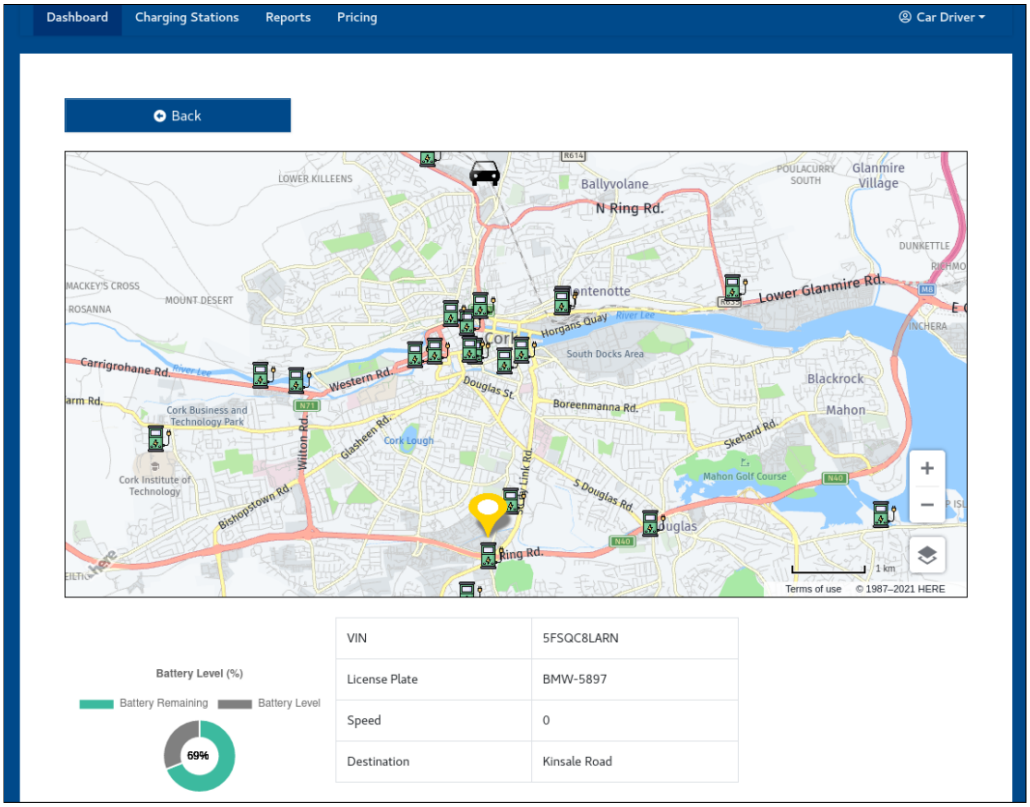


Figure 1.7: Dashboard for vehicle driver

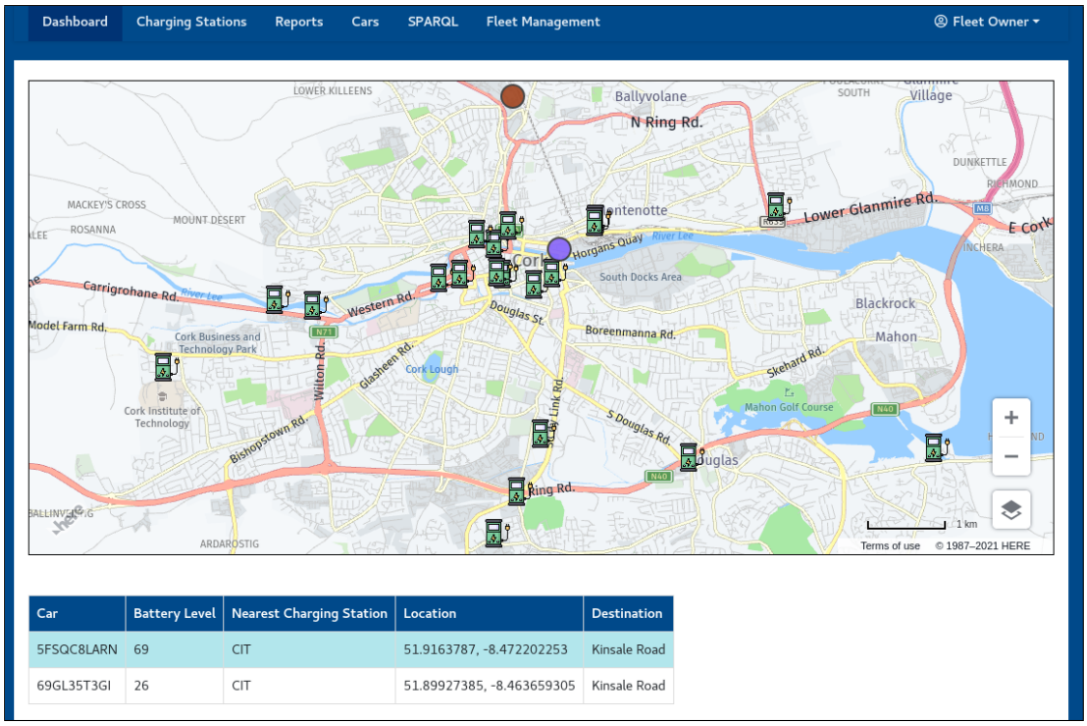
Vehicle Driver Dashboard. The Vehicle Driver view has four tabs: Dashboard, Charging Stations, Reports and Pricing. Figure 1.7 shows the Dashboard view. The vehicle driver can see the details related to the vehicle. For example, the battery level, location of nearest charging station, current location, speed etc.

Fleet Owner Dashboard. The Fleet Owner view has six tabs: Dashboard, Charging Stations, Reports, Cars, SPARQL and Fleet Management. Figure 1.8 shows the Dashboard view. The Fleet Owner can see vehicles and charging stations marked on the map, along with vehicle details in the dashboard. In the Reports tab, analytics (average speed, average battery level) will be displayed (if already calculated), and/or we can calculate the average values and the data can be re-ingested. Fleet Management tab is used to link a vehicle driver to the VIN. This linking is required so that the vehicle driver can log in to see the details.

EV Infrastructure Provider Dashboard. The EV Infrastructure Provider view has five tabs: Dashboard, Charging Stations, Reports, SPARQL and Pricing. Figure 1.9 shows the Dashboard view. The EV Infrastructure Provider can view the status of the EV charging stations. Also, the user can view analytics of average frequency of EV charging station being full/empty, which is overlayed as a heatmap on the map.

Raw Data Access

The data received from vehicles are stored as JSON files and RDF triple stores. The Web UI will facilitate accessing this raw data from storage for each user by adding the user credentials to the



Car	Battery Level	Nearest Charging Station	Location	Destination
5FSQC8LARN	69	CIT	51.9163787, -8.472202253	Kinsale Road
69GL35T3GI	26	CIT	51.89927385, -8.463659305	Kinsale Road

Figure 1.8: Dashboard for Fleet Owner

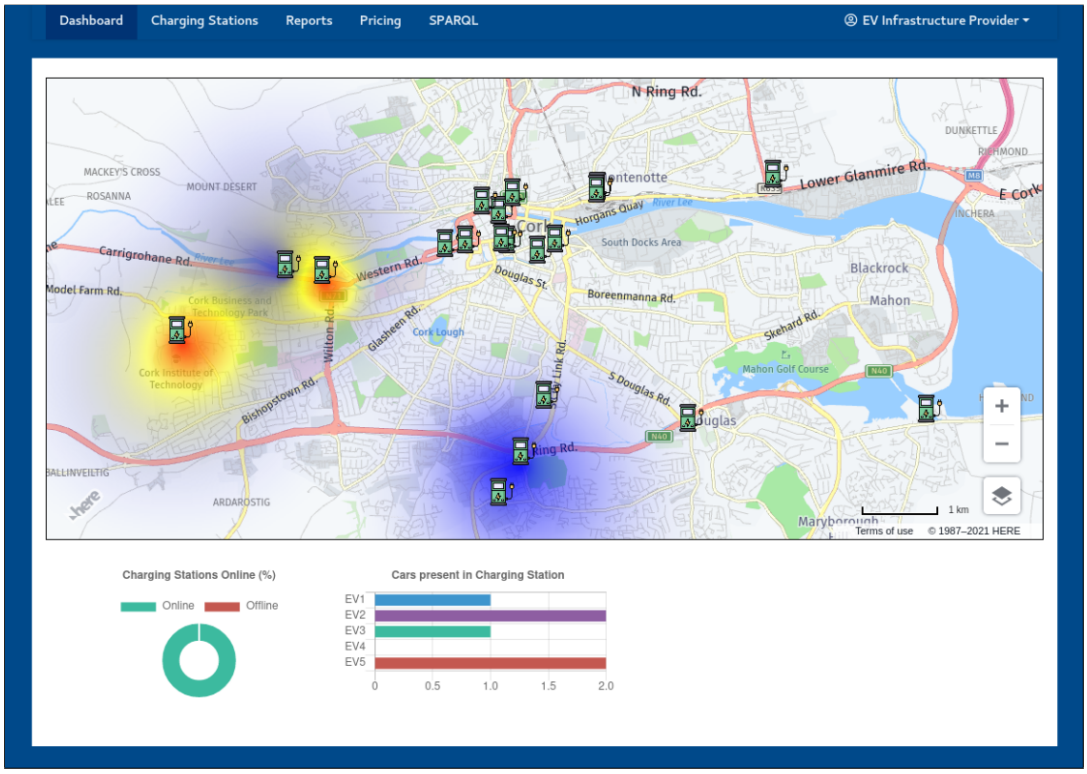


Figure 1.9: Dashboard for EV Infrastructure Provider

query thus implementing access control, authorization and restrictions. The data available depends on the privacy of the data and access rights of the user.

1.5 Policy Engine

This section gives an overview of the Policy Engine developed by the academic partners, which is detailed in Deliverables D3.3 “First Version of Policy Specification Language and Model” and D3.4 “Final Tools for the Governance Framework”, and explains the extension of its functionality that enables it to be integrated into the UC1 MOSAICrOWN platform.

1.5.1 Overview of Tools

The Policy Engine is responsible for parsing the user defined policies and checking whether an access request for a given piece of subject data is permitted or denied.
For UC1, the Policy Engine was implemented as an access control mechanism, operating on requests from the Web UI to the data market. A subject in this context refers to the vehicle and related metadata. Access to data is validated through the policy linked to the vehicles metadata. The policy defines the permitted data points and the context in which they can be accessed, including its intended purpose, the action to be performed, and user making the request. All access requests to data in the data market are governed by the Policy Engine. The modifications of the different components of the Policy Engine are described next.

1.5.2 Modification of Tools

Containerization

The Policy Engine was containerized using Docker and deployed and managed with the other core components of the platform. This modification enables smoother integration and management of the tool with other components, as well as a portable and reproducible deployment of all components of the platform. Deploying the Policy Engine in a containerized environment also allowed for greater control in the definition of communication mechanisms with other related components of the platform.

API

For UC1, communication between the Policy Engine and Web UI is made through HTTP requests. The API was developed using Flask, which is a Python based web framework. The primary role of this API is to receive and parse incoming HTTP requests, passing the incoming request to the Policy Engine Front-end and Core in a compatible format. Flask is used to wrap the Policy Engine and expose it as an access control service within the UC1 MOSAICrOWN platform. The API provides two endpoints: a subject query, where a specific ID (subject) is defined, and subjectless queries, which only define predicates to query. Both of these endpoints implement a remote policy loading function to retrieve the subject’s linked policy data from the RDF metadata triple store, before passing the policy and query to the Policy Engine Core. If the Policy Engine Core permits the subject access request, data (specified by the SPARQL query) is returned to the requester, in this case the Web UI. Figure 1.10 illustrates the Policy Engine API data flow.

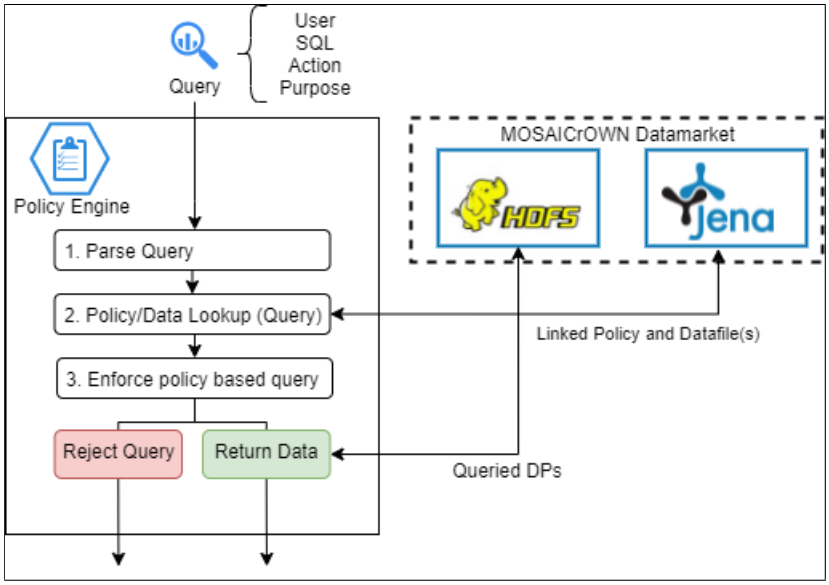


Figure 1.10: Policy Engine API data flow

```
{
"query": "SELECT ?vin ?seatLoc WHERE {?vin <http://dellemc.com
    :8080/icv/driverSeatLocation> ?seatLoc .}",
"type": "vehicle",
"user": "fleetmanager",
"action": "read",
"purpose": "statistical"
}
```

Figure 1.11: Parameters of a subjectless API endpoint query

Remote Policy loading

The Policy Engine was extended with functionality for loading a MOSAICrOWN Policy at request time. Using the Policy Engine's SPARQL query parsing, either one (ID) or many (subjectless) identifiers are extracted (Figure 1.11). Using these ID(s), a request is made to the RDF metadata triple store requesting a policy for each ID. These policies are parsed and loaded into an in-memory RDF graph before being passed to the Policy Engine Core with the data request.

1.6 Encrypted File System

The encrypted file system provides UC1 with data wrapping capabilities. The data market filter component can send data for data wrapping, if encryption on that data is desired. The encrypted file system utilizes two tools developed by MOSAICrOWN consortium partners; namely, the FreyaFS encrypted file system which leverages the aesmix all-or-nothing transform Mix&Slice encryption library which were detailed in Deliverable D4.1 "First Version of Encryption-based Protection Tools" and D4.3 "Final Encryption-based Techniques". This section details the work carried out to extend the functionality offered by these tools.

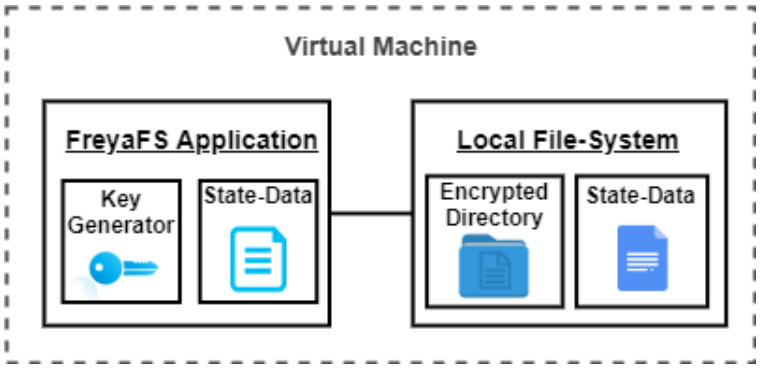


Figure 1.12: Architecture of initial FreyaFS utility

1.6.1 Overview of Tools

The FreyaFS encrypted file system provides the novel encryption of aesmix seamlessly and transparently by hooking in directly to the operating system's file system operations as an intermediary. FreyaFS uses a FUSE (file system user space) mount point to mount a local directory. Upon activation, FreyaFS ties up the current terminal as it runs as a service. Files created within, or moved into, the mounted directory are seamlessly encrypted using aesmix. This work addresses the requirements for centralized key management infrastructure (REQ-UC1-AC3) compression and encryption (REQ-UC1-DI6), protection at rest and in transfer (REQ-UC1-DM5).
The all-or-nothing-transform (AONT) Mix&Slice encryption mode creates a fully bit-interdependent representation of the source data via mixing. This encrypted representation is also sliced into fragments. Every fragment is necessary to successfully decrypt, and even one missing fragment prevents the decryption.

1.6.2 Modification of Tools

This section covers the main technical outcomes of the development of the encrypted file system tools. Three primary modifications have been carried out: the containerization of the FreyaFS encrypted file system, the investigation of container state externalization, and the integration of external key management, which are explained in the remainder of the section. These extensions are inherently interoperable and form a novel extension to FreyaFS to meet the requirements of UC1.

Containerization of FreyaFS

Work was carried out on containerizing the FreyaFS utility. Instead of FreyaFS running as a service on the host, the process now runs in a container which can be flagged to run in the background. This means that many instances of the utility can easily be run at the same time on one machine, whic has as result a more graceful starting and stopping of the service. The directory which is mounted using the FreyaFS tool exists within a container volume. This approach circumvents the issue of the containerized application itself losing state upon the container restarting.
The containerization of FreyaFS changes the method by which files are accessed. In the original version of FreyaFS, other use-case one components could read or write data directly into the FreyaFS mounted directory which was on a host's local file system. The method of file access for the containerized FreyaFS tool utilizes Docker container storage interface (CSI) plugins to enable

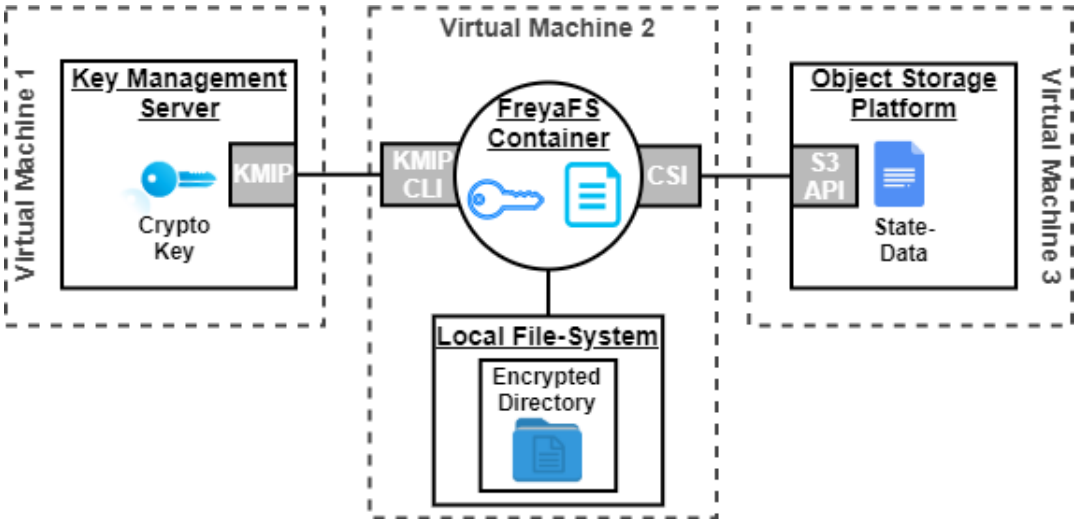


Figure 1.13: Architecture of modified FreyaFS utility

a direct pipeline from UC1 components, such as the data market filter to the encrypted file system.

Externalizing Container State

Exploratory work to build upon the utilization of container volumes was carried out by leveraging an external object storage platform to facilitate stateful container migration by making state-essential application data (such as FreyaFS metadata files) remotely accessible to containers. This means that a container can migrate between hosts, access its externalized data from remote storage, and then resume operation from its most recent copy of the data. The exploratory work fed into the externalization of the entire FreyaFS mounted directory to remote storage, rather than using a local directory. The system saves the FreyaFS data within an S3 bucket. The S3 bucket is exposed to a containerized application using a common storage interface driver to mount it locally to appear as a standard container volume. EISI's open source 'REX-Ray Driver' is the common storage interface plugin used to implement this functionality.

Integration of KMIP

The FreyaFS utility was extended with an OASIS Key Management Interoperability Protocol client using PyKMIP, which allowed for the replacement of the integrated password generator in FreyaFS with a remote key management server that provides cryptographic keys and management functionality to FreyaFS, and more specifically the aesmix cryptography library running in the background. FreyaFS being written in the Python language meant the PyKMIP extension can be included directly into the code.
The KMIP functionality currently operates using the PyKMIP client integrated into FreyaFS and a PyKMIP test server, but any OASIS KMIP standard server is compatible with the system. The KMIP-enabled FreyaFS version is integrated into the containerized build, and the container communicates with an external KMIP server to fetch keys. Any KMIP compliant server could be used with this system. The primary motivation behind the addition of a KMIP client into FreyaFS and supporting external key management was to address key management requirements as well as improve the general security of the tools in UC1.

1.7 Summary

This chapter presented an overview of UC1 and linked technological approaches. It demonstrated how the results of the MOSAICrOWN can be applied to the development of an automotive data market, where confidential information can be traded in a safe and secure manner. In particular, it shows the value of integrating the Policy Engine, developed by the academic partners, to effectively control access to data indexed by a metadata knowledge graph. The main items of work detailed in this document were: the Automotive Tools, the Web UI, the wrapping of the Policy Engine, and the Encrypted File System.

The use case implementation has provided us with practical experience of how to build a flexible and extensible data market based on emerging technologies, such and knowledge bases, RDF based access policies, and dynamically encrypted file systems. This design will help form a blueprint for future customer solutions that provide a way to safely and securely monetize data in an open market.

2. Use Case 2 (MC): Tools for financial data markets

This chapter details the tool to satisfy Use Case 2 (UC2) focusing on transaction-level financial data and on the importance of data wrapping techniques for the final purpose of analytics and the extraction of business insights from the data itself. UC2 considers requirements in Deliverable D2.1 "Requirements from the Use Cases" and the tools developed are layered as below.

Presentation. UC2 was designed as a Web Application where each user can access to the platform uploading their dataset and generating analytics after data anonymization.

Application. The application layer manages the logic and the engine of the platform. It has three different components:

- *Identity layer* allows users to be authenticated (Identity Server).
- *Wrapping layer* analyzes the dataset provided and stored in the data layer and, according to that, defines and applies the wrapping techniques for each field.
- *Analytics layer* once data is anonymized the users can access a dedicated dashboard to generate insights based on analytics built on the anonymized data. The analysis provides aggregated results, so no granular information is presented. The user must use a specific data-upload template, related to different industry and markets.

Data. The users, once connected to the platform via a web browser, upload the dataset to be anonymized. The platform proposes wrapping techniques for each field (according to the policy regulation selected), which can be confirmed or overwritten by the user, the platform anonymizes the dataset.

This chapter covers the cloud-based platform that anonymizes Personally Identifiable Information (PII). The application is designed to answer different user needs detailed as two different user journeys. One journey is dedicated to the data anonymization and the other one focuses on generating analytics from the anonymized data, as detailed next.

User journey 1 (Data Wrapping and Policy Language). The user selects a policy in alignment to the type of data and/or regulations to be applied. Then, the platform runs a semantic analysis of the data provided, recognizing the data types. A semantic and data distribution of the dataset is uploaded by the user, and accordingly wrapping techniques are applied. The wrapping techniques can also be overridden directly by the user, in addition to the predefined wrapping techniques defined for each metadata in the policy. Once the file is anonymized, the user can download the fully anonymized dataset. During the upload phase, the user has the possibility to choose if the uploaded dataset can be considered inside the Analytics section or simply pass through the actual anonymization phase. If the uploaded

dataset respects a predefined template, the dataset will be made available in the Analytics section once anonymized for analytics visualization.

User journey 2 (Data Analysis). Once the anonymization and the User journey 1 is completed, the user can access the Analytics section which utilizes the anonymized data. The user can visualize the summarized dashboard as well as each analytics page giving a segmented view by:

- product
- channel
- card type
- category

The related evolution during the time can be visualized:

- monthly
- quarterly
- yearly

The dataset needs to respect a predefined template to match the types of analytics above (i.e., product, channel, card type, and category), and for the platform to be able to generate analytics.

2.1 Background

2.1.1 Overview

Organizations must develop effective data strategies and utilize privacy-enhancing techniques to meet regulation requirements and consumer expectations, and to continue to innovate. Data have the potential to fuel innovation, but only if data practices are held to the high standards that customers and partners deserve with respect to the privacy regulations. The different privacy regulations are creating both challenges and expectations for innovation, differentiation as well as an opportunity to build trust with consumers. The regulations are also constraining market players to build privacy-by-design into standard business processes as well as leveraging best practice data management standards, like data minimization, to enable continued innovation. The adoption of data anonymization expands the amount of data we can use for analytics. This must be done allowing organizations to include in the source dataset not only current customers, but also customers who left the organization in the previous years. Anonymization increases data reliability of the insights provided, particularly the ones related to risks and frauds.

Context in MOSAICrOWN

The main focus of this chapter is within Work Package 2 (WP2) of MOSAICrOWN, and describes the solution developed in the context of UC2. The solutions are based on result of works and findings from Work Packages 3-5 and the focus is on preserving confidentiality of data for analytical purposes as per UC2 scope of work. This chapter relates to WP2 focusing on defining requirements of the different use cases and captures the final versions produced.

UC2: Background on core technologies

UC2 considers several key features, such as the recognition of the different level of Personally Identifiable Information (PII) in financial data. Also, UC2 considers the application of adequate wrapping techniques according to the recognized data type and the possibility for the user to adjust both the data type and wrapping technique to be applied. Data governance, wrapping, and sanitization are key components of UC2.

2.2 Solutions

Functionality and goals

UC2 is a cloud-based platform, where each user can upload a data file for anonymization and later generates analytics from the anonymized dataset.
The user selects a policy which is the interpretation of privacy regulations. The policy defines a set of wrapping techniques to apply to generate an anonymized dataset compliant to regulation. Then, the platform runs a semantic analysis of the data provided, recognizing data types, semantic and data distribution of the dataset uploaded by the user; accordingly wrapping techniques to be applied are proposed to the user. Different wrapping techniques are proposed by the platform according to the policy selected, the data type, and the data distribution reported in each field; nevertheless the user can still customize the wrapping techniques to be applied. The final output, the anonymized file, can be downloaded by the user with the fully anonymized dataset.
On top of the application and customization of anonymization techniques on data, the data are also processed to generate analytics.

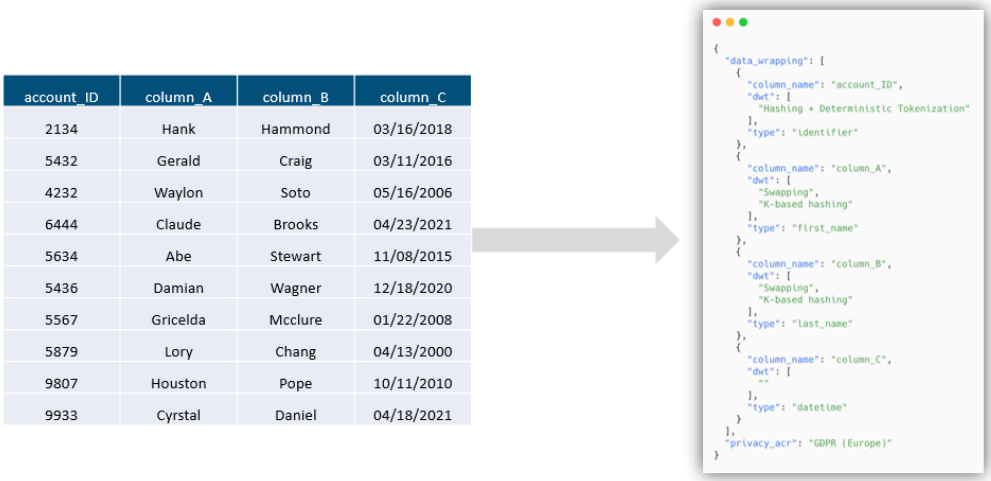
Policies

Figure 2.1 is an example of sample dataset with the policy file defined for it by the pilot. A policy, which is a formalization of the privacy regulation in a technical language, provides a schema combining data and anonymization techniques based on privacy requirements defined in the regulations. For each data type, the policy proposes a selected wrapping technique applicable. Once the dataset analysis is completed, the user can also select a different wrapping techniques from the one proposed by the policy, to anonymize the related data field. The platform can manage different privacy regulations and, for each of them, a configuration file, in Jason format, is created and uploaded into the system. This allows the platform to maintain the policies stored according with the evolution of the regulation over the time.

2.3 Architecture and Components

2.3.1 Client-Server Model

The solution architecture is based on three different tiers: presentation, application, and data as shown in Figure 2.2.
In the presentation tier, there is the web application layer, utilized by users to access the platform, upload their datasets to be anonymized and go through the analytics section. This tier communicates with the application tier and allows users to identify themselves and be authenticated (Identity layer), upload datasets, choose anonymization techniques, receive datasets in anonymized



account_ID	column_A	column_B	column_C
2134	Hank	Hammond	03/16/2018
5432	Gerald	Craig	03/11/2016
4232	Waylon	Soto	05/16/2006
6444	Claude	Brooks	04/23/2021
5634	Abe	Stewart	11/08/2015
5436	Damian	Wagner	12/18/2020
5567	Gricelda	Mcclure	01/22/2008
5879	Lory	Chang	04/13/2000
9807	Houston	Pope	10/11/2010
9933	Cyrstal	Daniel	04/18/2021


```
{
  "data_wrapping": [
    {
      "column_name": "account_ID",
      "dwt": [
        "Hashing + Deterministic Tokenization"
      ],
      "type": "identifier"
    },
    {
      "column_name": "column_A",
      "dwt": [
        "Swapping",
        "K-based hashing"
      ],
      "type": "first_name"
    },
    {
      "column_name": "column_B",
      "dwt": [
        "Swapping",
        "K-based hashing"
      ],
      "type": "last_name"
    },
    {
      "column_name": "column_C",
      "dwt": [
        ""
      ],
      "type": "datetime"
    }
  ],
  "privacy_acr": "GDPR (Europe)"
}
```


Figure 2.1: Configuration file example

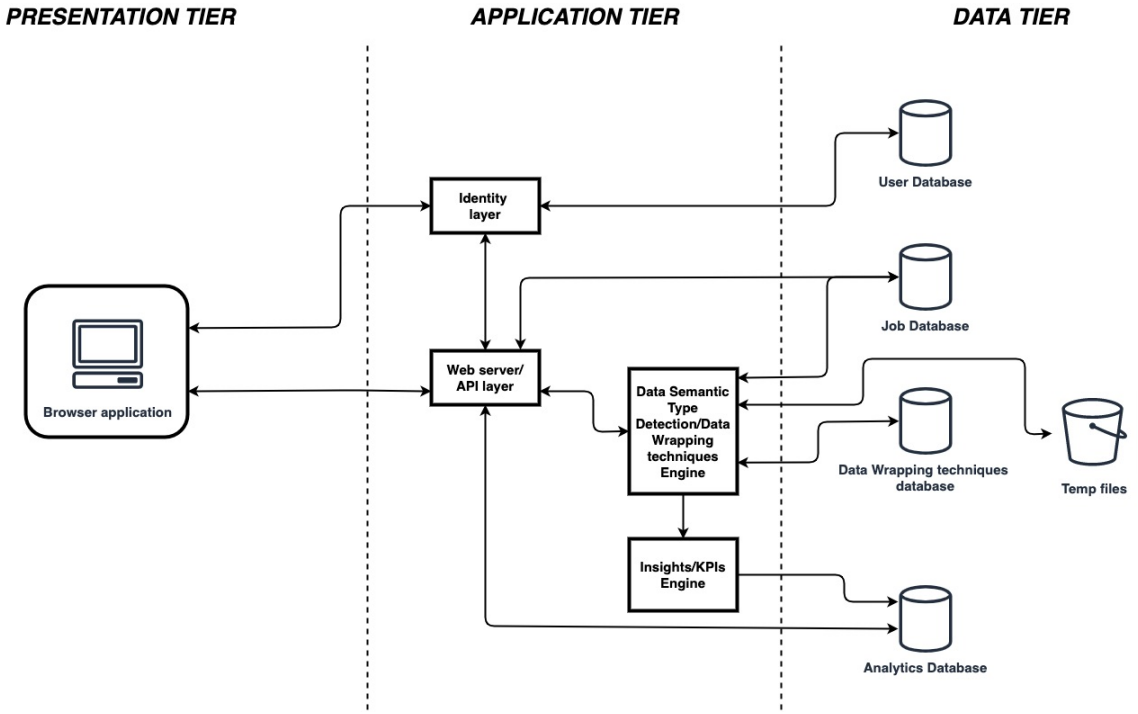


Figure 2.2: Three tier architecture

format and see aggregated values available in the Analytics (Analytics KPIs) and insights (API layer). In the application layer there are:

- Identity layer
- Web server/API layer
- Data semantic/data wrapping Engine (DSDW Engine)
- Insights/KPIs Engine

The Identity layer and the Insights/KPIs Engine have been developed in C# while Web server/API

layer and DSDW Engine have been developed using Python and Flask framework (a web framework for Python) and then placed inside a docker. The Identity layer is used to authenticate users providing them a token with a validity time, enabling them to use all the services offered by the platform until the valid time expires. The Web server/API layer serves the contents to be shown on the web interface, communicates with the DSDW Engine to perform the anonymization processes and with Insights/KPIs Engine to feed the Analytics section. The DSDW Engine has two tasks: *i)* semantically recognize the content of the dataset, *ii)* perform the anonymization processes requested by the user.
In the data tier, all the required data structures needed to the correct behavior of the platform. In details:

- Job database contains the processes (jobs) that perform data semantic detection and data wrapping techniques application. These jobs have a duration that varies depending on the number of rows and columns in the dataset.
- Data Wrapping techniques database contains the set of wrapping techniques associated with the various semantic types divided by privacy policies (e.g., GDPR, Brazilian privacy regulation LGPD).
- User database contains the set of authorized users able to access the platform.
- Analytics database contains all the data to be used inside the Analytics section.
- Temp files bucket is a bucket containing all the temporary files produced by the execution of the data semantic detection and data wrapping techniques application jobs. These temporary files are immediately deleted once they are considered no longer needed for the subsequent steps of the platform.

All databases inside the presentation tier are relational, with a predefined schema.

2.3.2 User Interface

The user interface has been developed in Angular (Angular is a platform and framework for building single-page client applications using HTML and TypeScript). They drives, by a dedicated workflow, users through the various steps in order to receive a dataset containing anonymized data according to the selected wrapping techniques. The UI interacts with a backend layer through authenticated REST API calls (the authentication token provided by the Identity server must be inserted in every call made after login). Once logged into the platform, the user is asked for the policy to be applied. The user is also asked to select the user journey, which is either only the dataset anonymization or the dataset anonymization and the generation of analytics, as shown in Figure 2.3. After the policy and the user journey have been selected, the platform asks the user to upload the dataset (the expected file types are xls, xlsx, csv). If the user selects data anonymization and analytics generation, the dataset format needs to match the analytics structure. If the dataset uploaded does not match this format, the user receives an error message, as shown in Figure 2.4, and the option to download the correct template for the dataset uploaded. Once the analysis of the loaded dataset is finished, a sample of the data is shown and, for each column, there is the semantic type of the dataset columns that DSDW Engine has predicted. However, as shown in Figure 2.5, the user has the possibility to change the semantic type of each column by choosing one contained into the policy previously selected. Figure 2.6 visualizes how the platform shows,

for each column, the set of the data wrapping techniques that can be executed on that particular semantic type. Once the user has chosen the data wrapping techniques to be applied, the platform allows the download of the anonymized dataset as shown in Figure 2.7.

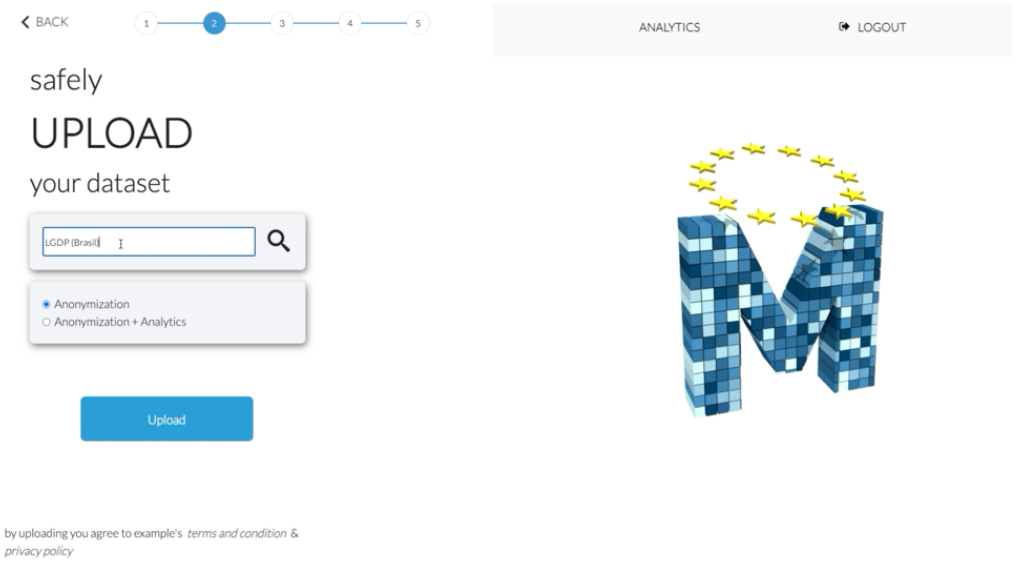


Figure 2.3: Select privacy policy and user journey

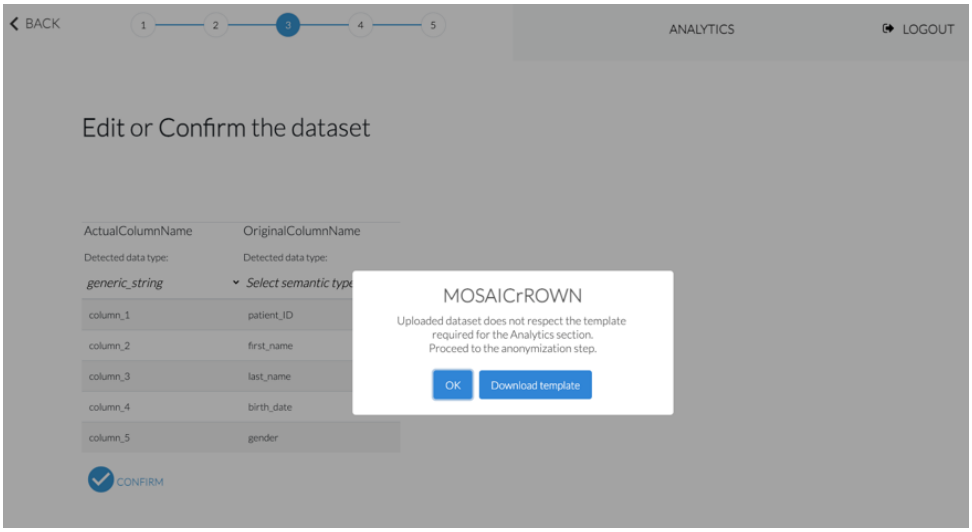


Figure 2.4: Error message and data upload structure template

Analytics Section

The user has the possibility to go directly to the Analytics section using the link on the top right corner of the platform near logout button. The Summary tab, Figure 2.8, shows some aggregated KPIs with respect to transactions, average spend per ticket, MoM/QoQ/YoY values based on the last month/quarter/year of data in the database and a section containing several pie charts showing some metrics divided by channel, by products and by industry.

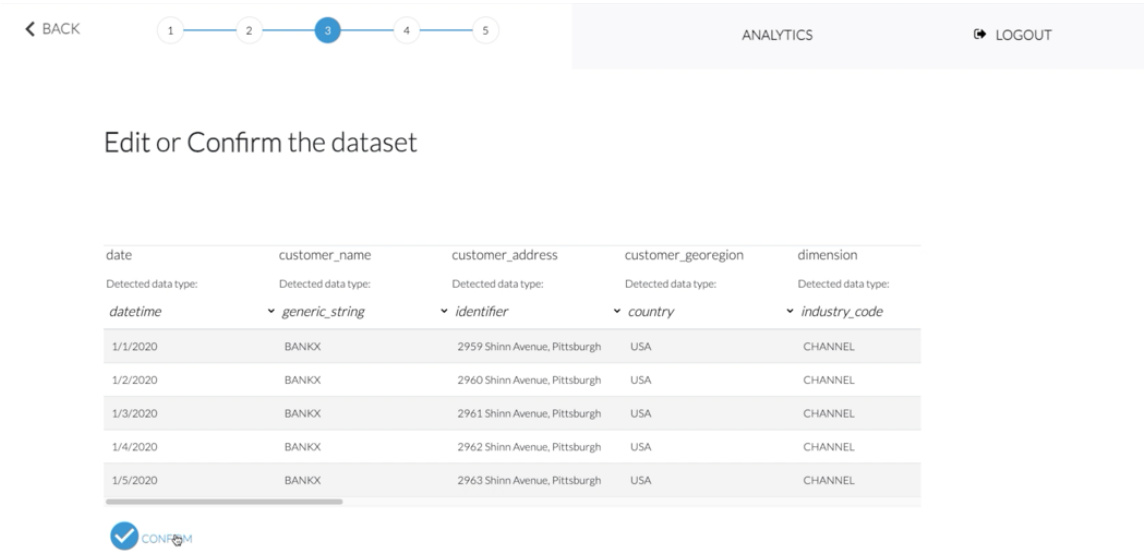


Figure 2.5: Data semantic type detection

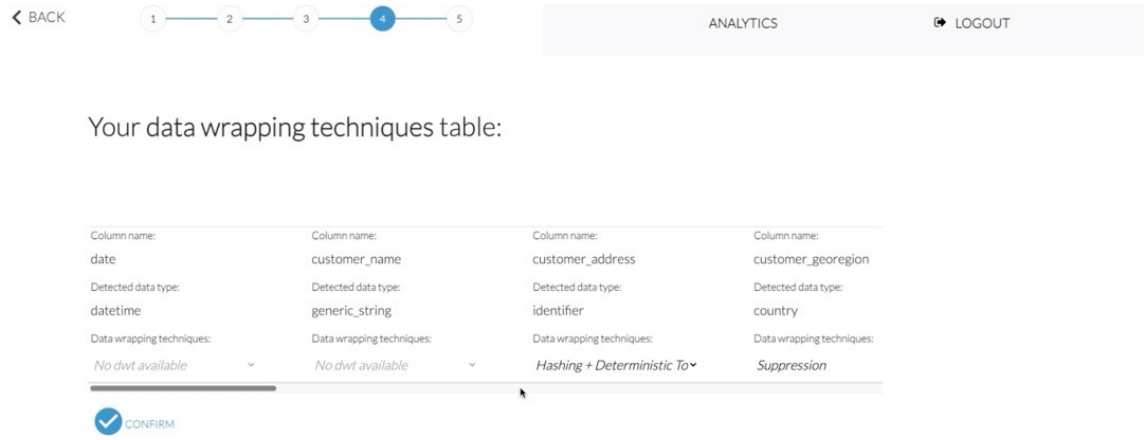


Figure 2.6: Data wrapping techniques

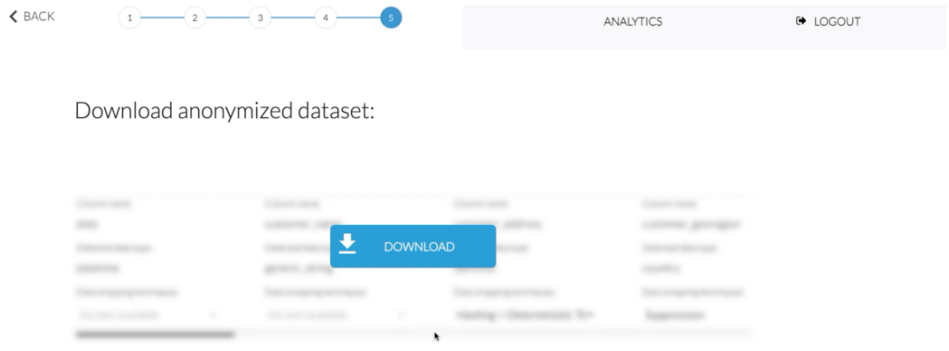


Figure 2.7: Download anonymized dataset

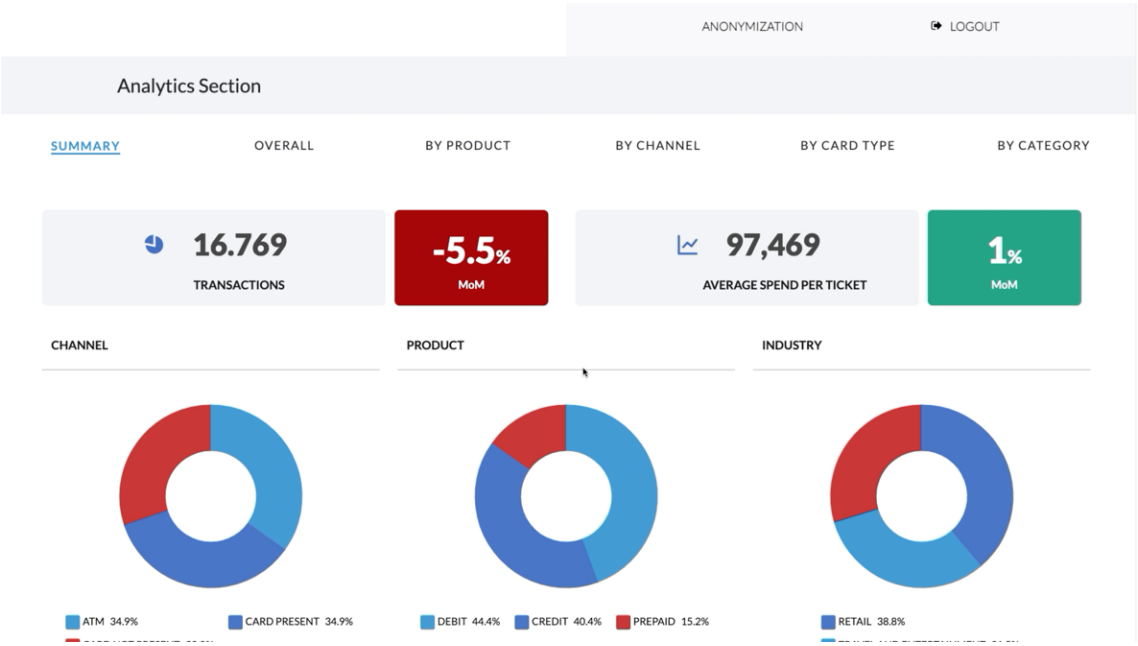


Figure 2.8: Analytics section - Summary tab

The Overall section, Figures 2.9 and 2.10, shows the set of all aggregated data with the possibility of: *i)* reducing the reference time interval (monthly, quarterly, yearly), and *ii)* selecting the index on which the underlying insights are calculated (spend, transactions, average ticket size). The graph in Figure 2.10 shows the temporal trend of the selected index.

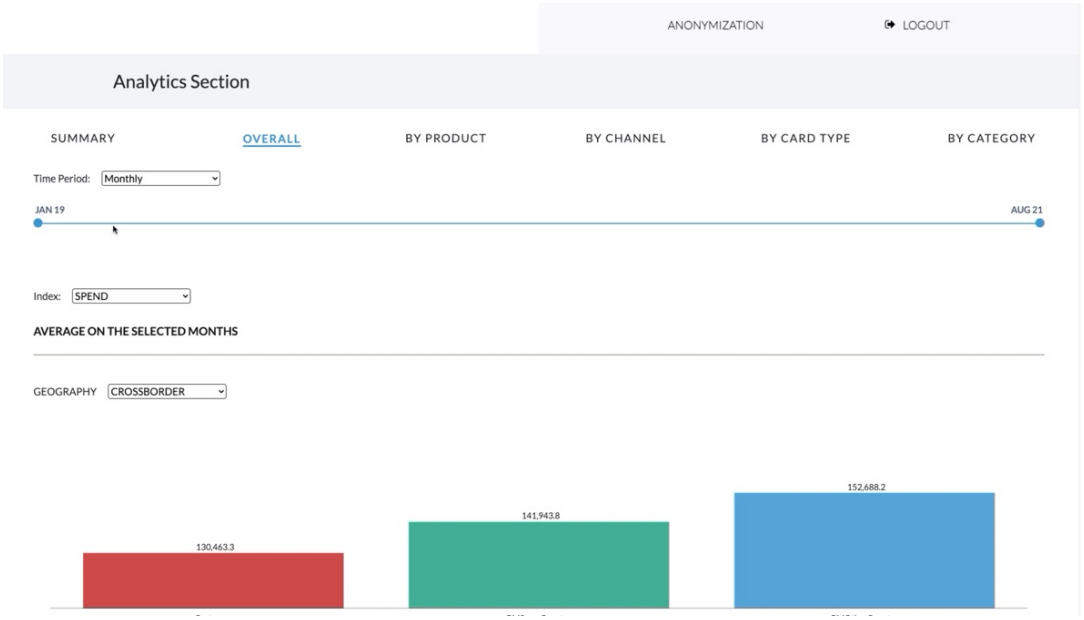


Figure 2.9: Analytics section - Overall view (1/2)

In the tab By Product in Figure 2.11 (and also in all the following sections), different KPIs are shown, depending on the section itself, with the same possibility of changing the time interval and the reference index.

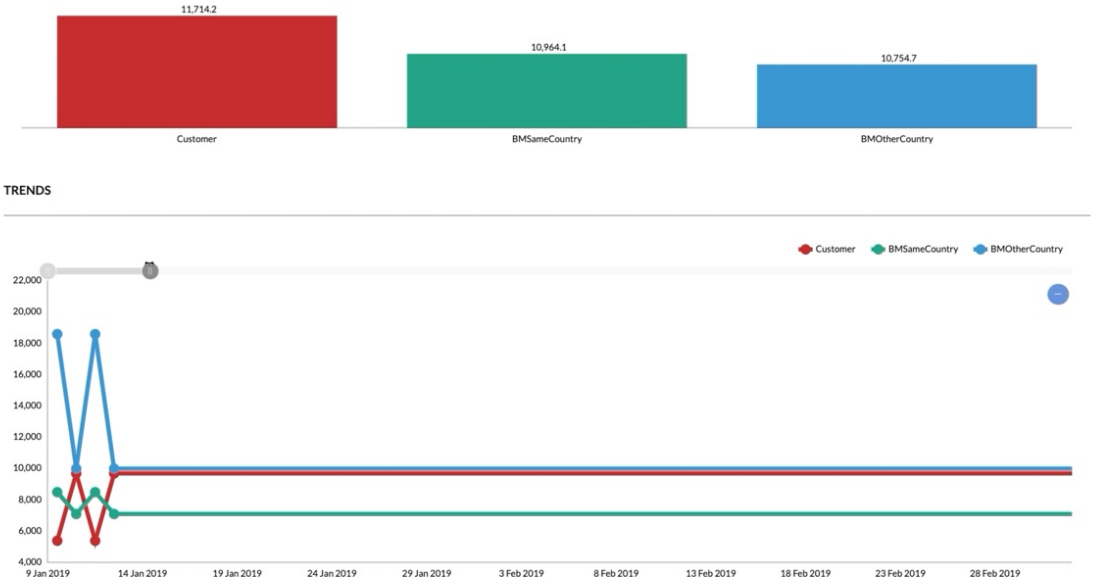


Figure 2.10: Analytics section - Overall view (2/2)

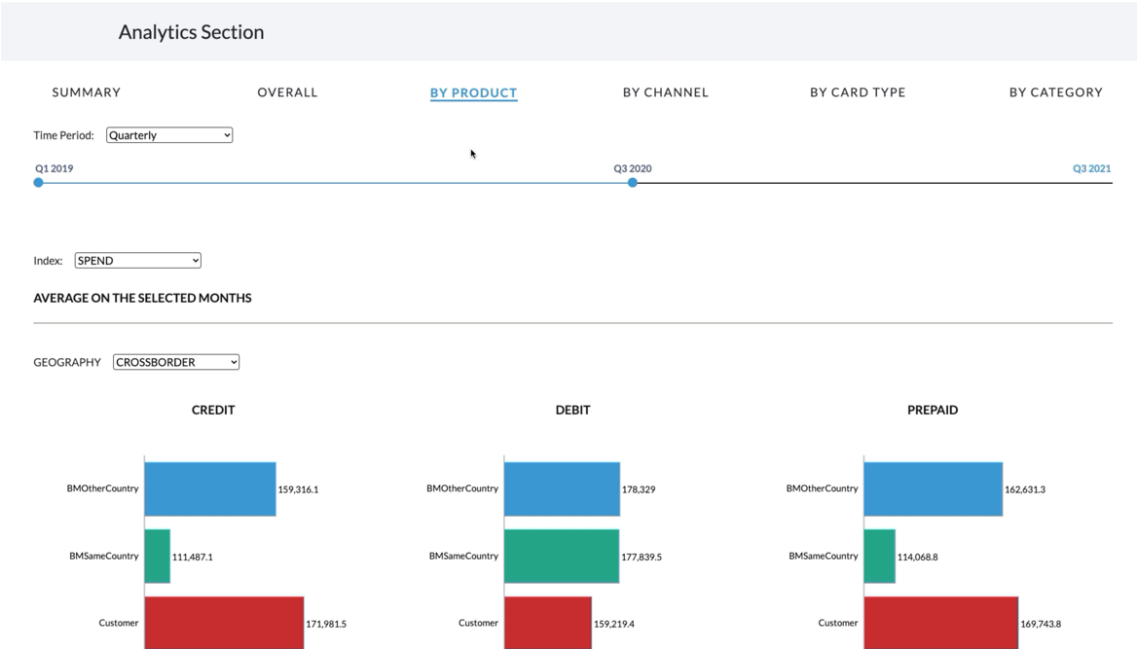


Figure 2.11: Analytics section - By product

2.3.3 Application Review

The proof-of-concept consists of three applications, one for each functionality of the platform:

- The core of the solution, which is based on DSDW Engine and its functionalities. Once the dataset is uploaded by the user, the same dataset is saved in the bucket named with a

unique alphanumeric identifier. A new job is created in the job table associated with that temporary file. The DSDW Engine takes over the job and performs a pre-processing phase on the received dataset based on tokenization/embedding extraction processes (required for the next phases).

- After this phase the DSDW Engine computes, for each column of the dataset, the probability that the column belongs to a particular semantic type according to its content. The semantic type with the highest probability is selected by default but the user can modify it if needed. Once the semantic type of each column has been identified, the job status is updated to allow the interface to show the result of the analysis. Once the user has selected for each column the data wrapping technique to be applied, the DSDW Engine applies the techniques on the original dataset creating a new temporary file (containing the data in anonymized form). During the upload phase, if the user has selected the request to upload the dataset in the Analytics section, the anonymized dataset will be uploaded to the analytics database. This file is then made available to the user who has the possibility to download it.

- Clicking on the Analytics link, the user has the possibility to browse the Analytics section containing the aggregated KPIs of all the datasets uploaded up to that moment, divided by sections of interest (By Product, By Channel, By Card type, By Category).

2.3.4 Process Diagram

Figure 2.12 describes the overall process of data analysis and wrapping, starting from an unstructured dataset provided by the user. The process is fully asynchronous (mandatory considering the amount of data that could be involved in this data transformation). The user can run the process and retrieve the related job monitoring the execution. The same approach is used for the application of the selected data wrapping techniques considering also the possibility to upload the anonymized dataset on analytics database if the dataset format is the format required by the platform.

2.3.5 Sequence Process

Figure 2.13 shows the different sequences we defined in the pilot, and the different components involved related to the anonymization process. Figure 2.14 demonstrates the different sequences we defined, and the different components involved in the different layers of the platform related to the analytics process. Regarding the Analytics section of the platform, the API requires the following parameters: Section, StartDate and EndDate. Section refers to the type of analytics – i.e., summary, overall, by card type, by category, by channel and by product; as shown as tabs in, e.g., Figure 2.8. StartDate and EndDate define the requested time frame for the analytics result. The API layer provides the data to populate the requested fields in terms of dimensions, KPIs and graphs.

2.4 Summary

This chapter outlined the final prototype for the UC2: data sharing and analysis through data anonymization techniques and compliance and the use of the anonymized data to generate analytics. Specifically, we illustrated how UC2 is realized via a cloud-based storage solution compatible with any public cloud provider. Demo access is provided via a central web front-end. We see the

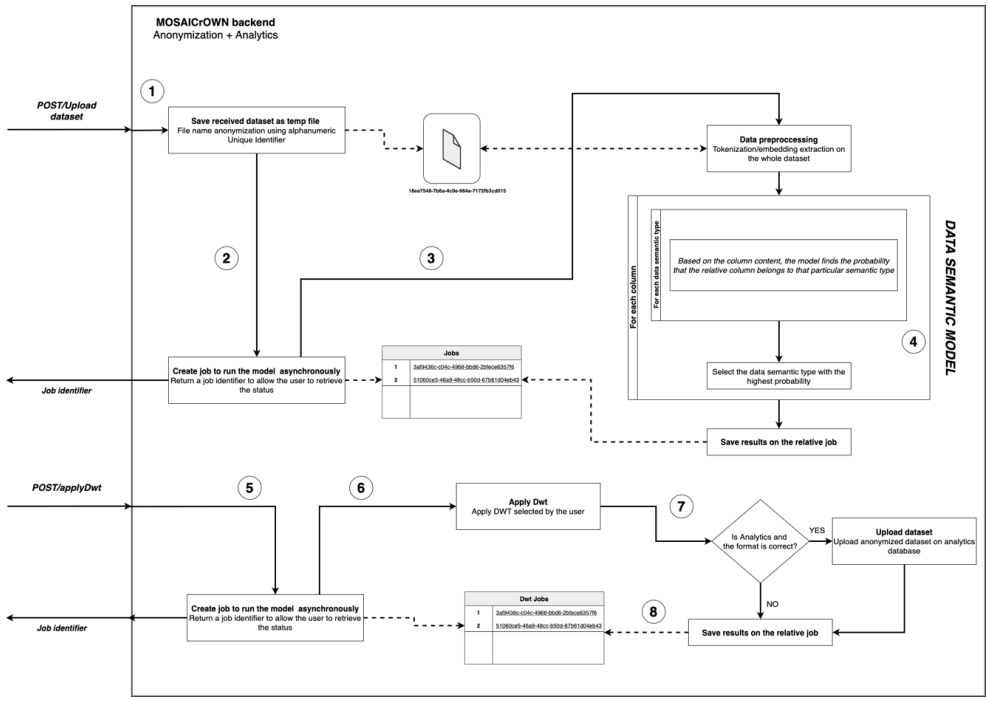


Figure 2.12: Process diagram

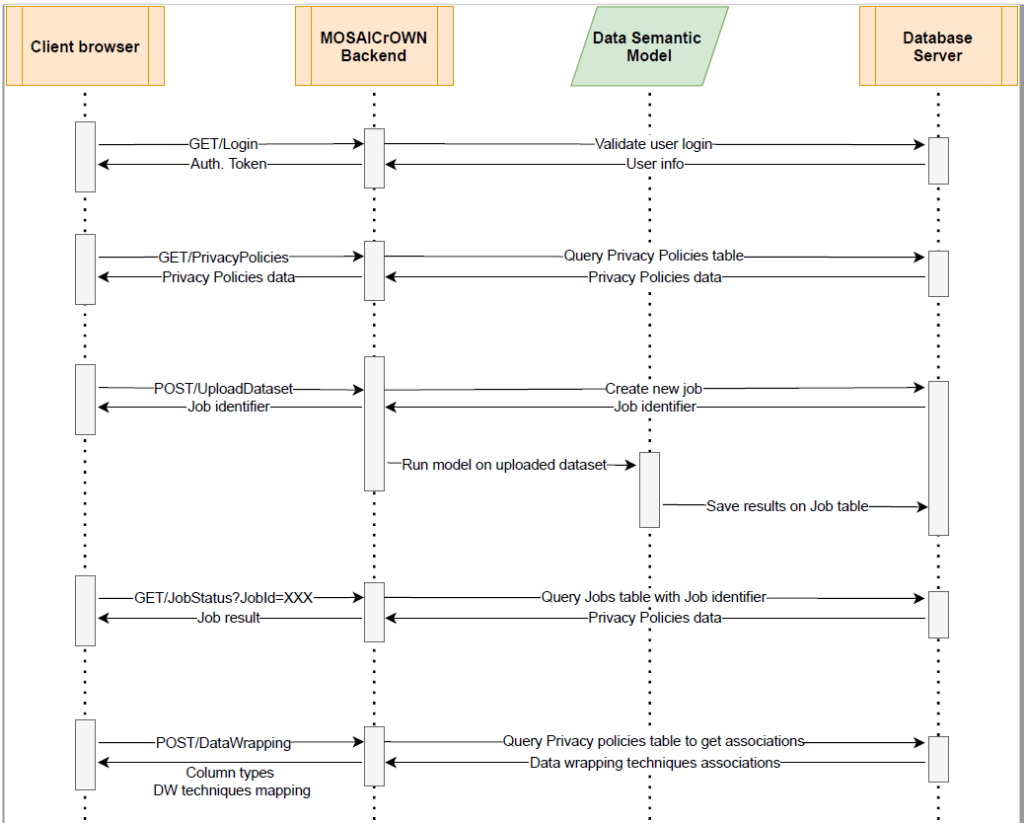


Figure 2.13: Sequence process - Anonymization

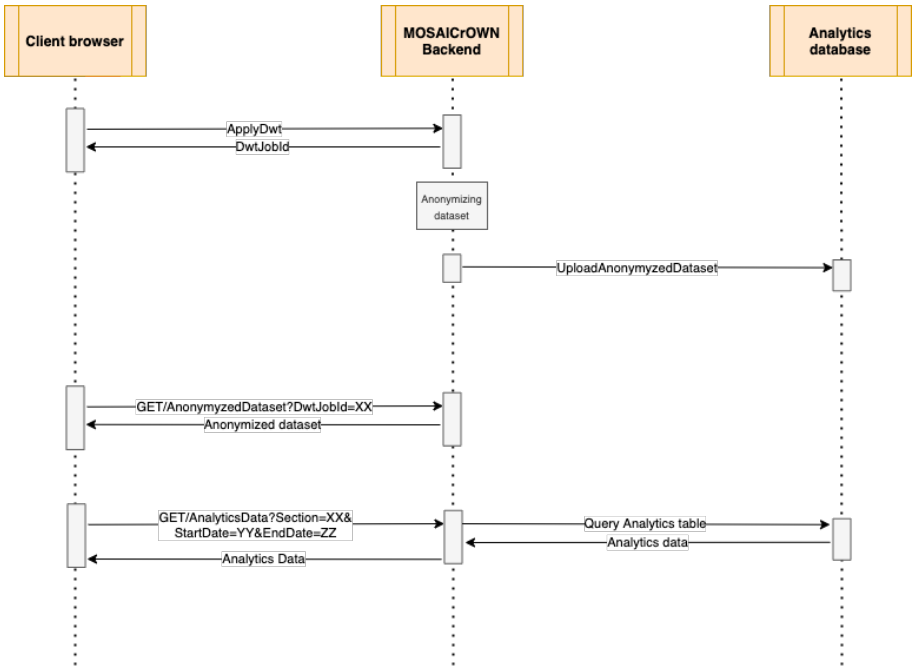


Figure 2.14: Sequence process - Analytics

presented prototype as an important milestone to enable significantly more secured information sharing between multiple parties with the guarantee to preserve PII information. UC2 prototype has been designed for the FSI (Financial Services Sector) industry, and particularly digital payment analysis. Nevertheless the platform is industry-independent and can manage any type of data to generate insights.

3. Use Case 3 (SAP SE): Privacy-preserving tools for cloud-based data markets

3.1 Introduction

This chapter covers Use Case 3 (UC3), a cloud-based data market for privacy-preserving analytics. UC3 considers two parties in a business-to-business scenario that want to compute privacy-preserving statistics over their joint data. Data sanitization that permits meaningful insights is the main technological issue of this use case. Differential privacy, a privacy notion satisfied by randomized algorithms that, e.g., perturb an analytical output, is the main technology applied in this use case. However, it can be augmented with different wrapping techniques investigated in WP4, which provide complementary protection, e.g., encryption at rest. The sanitization is ensured at different points throughout the data life-cycle, covering all phases: ingestion, storage, and analytics. Figure 3.1 shows these different phases and where sanitization can be applied.

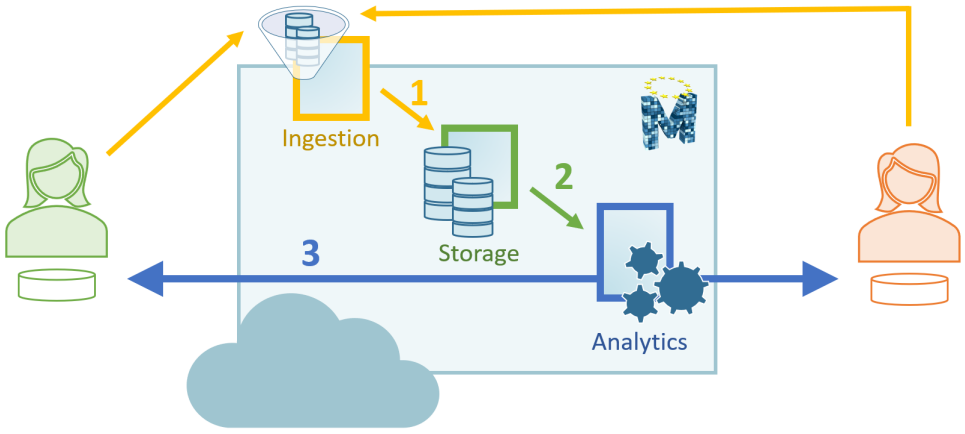


Figure 3.1: UC3 Overview

In option 1, the data is directly anonymized during ingestion, i.e., only anonymized data is stored. In option 2, the data is stored in plaintext (or encrypted) and only anonymized on demand (e.g., with regards to a chosen mechanism or analytical function). In option 3, the data is directly analyzed between the parties and they only learn the anonymized analytical result over their joint data. The tools for Use Case 3 cover the options mentioned above: The anonymization tool DPtool, detailed in Section 3.3.1, covers option 1 as well as 2. The privacy quantification and anonymization tool MIA, described in Section 3.3.2, also covers option 1 and 2. The secure computation tool DPsc, detailed in Section 3.3.3, covers option 3.
The remainder of this chapter is organized as follows. First, we provide preliminaries and technological background for differential privacy as well as secure computation in Section 3.2. Then, we provide tools to satisfy the requirements of UC3, namely, DPtool, MIA and DPsc, in Section 3.3.

3.2 Background

In this section, the required background for the tools of this use case is briefly presented. First, Section 3.2.1 describes anonymization mechanisms for differential privacy. Then, Section 3.2.2 details cryptographic tools for secure computation.

3.2.1 Background on Differential Privacy

In the following, we recall some preliminaries and definitions for differential privacy. Further details, can be found in deliverables D5.1 "First Version of Data Sanitisation Tools", D5.2 "First Report on Privacy Metrics and Data Sanitisation", and D5.3 "Final Report on Privacy Metrics, Risks, and Utility". We model a data set $D = \{d_0, \ldots, d_{n-1}\}$ as n elements from a data domain $\mathcal{U}$. A *neighboring* data set D' can be created from D by removing or adding an element.

Differential Privacy

Definition 1 (Differential Privacy)**.** *A mechanism $\mathcal{M}$ satisfies (ε, δ)-differential privacy, where $\varepsilon, \delta \geq 0$, if for all neighboring data sets D and D', and all subsets S of $Range(\mathcal{M})$*

$$Pr[\mathcal{M}(D) \in S] \leq \exp(\varepsilon) \cdot Pr[\mathcal{M}(D') \in S] + \delta,$$

where $Range(\mathcal{M})$ denotes the set of all possible outputs of mechanism $\mathcal{M}$.

Note that the definition above does not set any bounds on an adversary. However, due to our use of secure computation, we also require a definition with computationally bounded adversaries. Mironov et al. [MPRV09] define computationally indistinguishable differential privacy (IND-CDP) for two-party computation (2PC) with computationally bounded parties. He et al. [HMFS17] adapt the definition of Mironov et al. for parties A, B with data sets D_A, D_B, privacy parameters $\varepsilon_A, \varepsilon_B$ and security parameter λ. Furthermore, $\texttt{VIEW}_A^{\Pi}$ denotes the view of A during the execution of protocol Π.

Definition 2 (IND-CDP-2PC)**.** *A two-party protocol Π for computing function f satisfies $(\varepsilon_A(\lambda), \varepsilon_B(\lambda))$-indistinguishable computationally differential privacy (IND-CDP-2PC) if $\mathit{VIEW}_A^{\Pi}(D_A, \cdot)$ satisfies $\varepsilon_B(\lambda)$-IND-CPA, i.e., for any probabilistic polynomial-time (in λ) adversary $\mathcal{A}$, for any neighboring data sets (D_B, D_B')*

$$\begin{aligned} &Pr[\mathcal{A}(\mathit{VIEW}_A^{\Pi}(D_A, D_B)) = 1] \\ \leq &\exp(\varepsilon_B) \cdot Pr[\mathcal{A}(\mathit{VIEW}_A^{\Pi}(D_A, D_B')) = 1] + negl(\lambda). \end{aligned}$$

Likewise for B's view for any neighbors (D_A, D_A') and ε_A.

For convenience, we use $\varepsilon = \varepsilon_A = \varepsilon_B$.

Mechanisms

Randomized algorithms, called mechanisms, are required to satisfy Definition 1.
The *Laplace mechanism* [DR14] works by adding noise sampled from the Laplace distribution to a function evaluation.

Definition 3 (Laplace Mechanism)**.** *The* Laplace mechanism *for function* $f : \mathcal{U}^n \to \mathbb{R}$ *which has* l_1*-sensitivity* $\Delta f = \max_{\forall D \simeq D'} |f(D) - f(D')|$*, releases*

$$f(D) + Laplace\,(\Delta f/\varepsilon)\,,$$

where $Laplace\,(b)$ *denotes a random variable from the Laplace distribution with scale b and density* $Laplace\,(x;b) = \frac{1}{2b}\exp\Big(-\frac{|x|}{b}\Big)$.

The Laplace mechanism satisfies $(\varepsilon, 0)$-DP, also called pure differential privacy. Similarly, the *geometric mechanism* [GRS12] adds noise from the symmetric[1] geometric distribution. The symmetric geometric distribution is a discrete version of the Laplace distribution, i.e., it operates over integers, and is especially suited for anonymizing integer counts.

Definition 4 (Geometric Mechanism)**.** *The* Geometric mechanism *for function* $f : \mathcal{U}^n \to \mathbb{R}$ *with* l_1*-sensitivity* 1 *releases*

$$f(D) + Geometric\,(\exp(-\varepsilon))\,,$$

where $Geometric\,(b)$ *denotes a random variable from the symmetric geometric distribution with scale b and density* $Geometric\,(x;b) = \frac{1-b}{1+b}b^{|x|}$.

The geometric mechanism satisfies $(\varepsilon, 0)$-DP. Another mechanism, called *Gauss mechanism* [DR14], uses noise from the Gauss distribution.

Definition 5 (Gauss Mechanism)**.** *The* Gauss mechanism *for function* $f : \mathcal{U}^n \to \mathbb{R}$ *which has* l_2*-sensitivity* $\Delta_2 f = \max_{\forall D \simeq D'} |f(D) - f(D')|_2$*, releases*

$$f(D) + N(0, \sigma^2),$$

where $\sigma \geq c\Delta_2 f/\varepsilon$ *with* $c^2 > 2\log(1.25\delta)$*, and* $N(0,\sigma^2)$ *denotes a random variable from the Gauss distribution with scale* σ^2 *and density* $N(x;0,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\exp\big(-\frac{1}{2}(\frac{x}{\sigma})^2\big)$.

The Gauss mechanism satisfies (ε, δ)-DP, also called approximate differential privacy. The *exponential mechanism* [MT07] computes utility scores for a fixed set $\mathcal{O}$ of possible outputs and probabilistically selects an element based on its score. The formal definition is according to [LLSY16].

Definition 6 (Exponential Mechanism)**.** *For any utility function* $u : (\mathcal{U}^n \times \mathcal{O}) \to \mathbb{R}$ *and a privacy parameter* ε*, the exponential mechanism* $\mathcal{M}_u^\varepsilon(D)$ *outputs* $o \in \mathcal{O}$ *with probability proportional to* $\exp\Big(\frac{\varepsilon u(D,o)}{2\Delta u}\Big)$*, where*

$$\Delta u = \max_{\forall o \in \mathcal{O}, D \simeq D'} \big|u(D,o) - u\big(D',o\big)\big|$$

is the sensitivity of the utility function. That is,

$$Pr[\mathcal{M}_u^\varepsilon(D) = o] = \frac{\exp\Big(\frac{\varepsilon u(D,o)}{2\Delta u}\Big)}{\sum_{o' \in \mathcal{O}} \exp\Big(\frac{\varepsilon u(D,o')}{2\Delta u}\Big)}.$$

The exponential mechanism satisfies $(\varepsilon, 0)$-DP. While the mechanisms based on directly adding noise are simpler than the exponential mechanism, the former cannot be applied on arbitrary outputs (e.g., strings) whereas the latter supports arbitrary output domains. Furthermore, the exponential mechanism provides better accuracy for certain tasks compared to the Laplace mechanism, which includes the median [BK20b, MG20].

[1] Also called *two-sided* or *double*.

Inverse Transform Sampling for Exponential Mechanism

The exponential mechanism provides selection probabilities for each output element, i.e., it induces a probability distribution over the output domain $\mathcal{O}$. To sample from this distribution, we use *inverse transform sampling* (ITS). ITS uses the uniform distribution to simulate any other distribution. First, one samples a random element $r \in (0.0, 1.0]$ uniformly. Then, one finds the output element $o_j \in \mathcal{O}$ such that $\sum_{i=1}^{j-1} \Pr[\mathcal{M}_u^{\varepsilon}(D) = o_i] \leq r < \sum_{i=1}^{j} \Pr[\mathcal{M}_u^{\varepsilon}(D) = o_i]$. Finally, one outputs o_j. To illustrate the intuition behind ITS, assume $\mathcal{O} = \{a, b\}$ where the selection probability for a is 10% and for b it is 90%. Now, say we fill an array L of size 100 with 10 elements a and 90 elements b and select an array index i at uniform random and output $L[i]$. Then, $L[i]$ is a with 10% probability and b with 90% probability.

Geo Indistinguishability

Geo indistinguishability [ABCP13] is a generalization of DP, developed for location privacy, i.e., it allows to randomize coordinates. It generalizes DP to arbitrary metrics between inputs.

Definition 7 (Geo Indistinguishability)**.** *A mechanism $\mathcal{M}$ satisfies ε-GI iff for all points x, x':*

$$d_P(\mathcal{M}(x), \mathcal{M}(x')) \leq \varepsilon d(x, x'),$$

where d is the Euclidean distance and d_P the distance between the distributions over $\mathcal{M}$ on inputs x, x', respectively.

A mechanism $\mathcal{M}$ satisfying GI for a two-dimensional point $y = (y_1, y_2)$ reports a noisy version of y instead, based on the planar (i.e., two-dimensional) Laplace distribution. Informally, $\mathcal{M}$ draws a uniformly random angle a in $[0, 2\pi)$, and draws a radius r by sampling from the Gamma distribution (which generalizes the Laplace distribution) with shape 2 and scale $1/\varepsilon$. Then, it outputs $(y_1 + r\sin(a), y_2 + r\cos(a))$.

3.2.2 Background on Secure Computation

Secure two-party computation allows two parties A and B with inputs x_A and x_B, respectively, to compute a function $y = f(x_A, x_B)$, such that the parties only learn y and nothing more. In other words, the parties jointly compute f but never learn each other's inputs.
We focus on semi-honest adversaries, also called passive adversaries, which do not deviate from a secure computation protocol, but try to learn as much as possible from the protocol execution. Malicious adversaries, also called active adversaries, on the other hand, try to alter the protocol execution and might provide malicious inputs.
The two main implementation paradigms for secure computation are *secret sharing* [Sha79, Bla79] and *garbled circuits* [Yao86]. The former considers arithmetic circuits (arithmetic operations, e.g., addition over integers) the latter Boolean circuits (logical operations, e.g., AND over bits).

Additive Secret Sharing

For additive secret sharing, we assume all values to be in $\mathbb{Z}_{2^b}$ and that all operations are performed modulo $\mathbb{Z}_{2^b}$. We consider two parties A and B, where party A holds a secret $s \in \mathbb{Z}_{2^b}$. The parties want to split the secret into two parts s_A, s_B, where A holds s_A and B holds s_B, such that both parts are required to reconstruct the secret.

Such splitting can be achieved by first drawing a uniformly random value $u \in \mathbb{Z}_{2^b}$, then setting $s_A = s - u$ and $s_B = u$ and distributing it to the corresponding parties. We write $\langle s \rangle = (s_A, s_B)$ to refer to the secret sharing of s.
Addition with sharings $\langle s \rangle, \langle r \rangle$ is straightforward as $\langle s + r \rangle = \langle s \rangle + \langle r \rangle = (s_A + r_A, s_B + r_B)$. Multiplication with a public value $t \in \mathbb{Z}_{2^b}$ is also simple as $\langle ts \rangle = t\langle s \rangle = (ts_A, ts_B)$. To multiply secret shares $\langle x \rangle$ and $\langle y \rangle$, however, requires further techniques such as *Beaver triples*. Beaver triples are shares $\langle a \rangle, \langle b \rangle, \langle c \rangle$ such that $c = a \cdot b$ [Bea91]. With such a triple, the parties now compute $\langle x - a \rangle, \langle y - b \rangle$, and reconstruct $\langle x - a \rangle$ as $\alpha = x - a$ and reconstruct $\langle y - b \rangle$ as $\beta = y - b$. Thus, they can express the multiplication of shares as $\langle xy \rangle = \langle c \rangle + \alpha\langle b \rangle + \beta\langle a \rangle - \alpha \cdot \beta$. Such Beaver triplets can be constructed via oblivious transfer [DSZ15], which we describe next.

Oblivious Transfer

Oblivious Transfer (OT) [Rab81] is a cryptographic primitive that is equivalent to secure computation [Kil88]. In other words, OT is sufficient to construct any kind of secure computation, making it a very powerful tool. Specifically, the functionality for 1-out-of-2 OT considers two parties: a sender with two secrets s_1, s_2 and a receiver. The receiver wants to learn one of these secrets but in such a way that the sender does not discover which one. A simple OT construction, based on the Diffie-Hellman key exchange [DH76], is given by Chou and Orlandi in [CO15]. While OT requires costly computations (asymmetric key cryptography and therefore, e.g., modular exponentiations), there are efficient constructions, so called OT extensions [Bea96, IKNP03], which use a few (base) OTs to build larger, logical OTs.

Garbled Circuits

A Boolean circuit consists of logical gates connected by wires which move the output from one gate to the input of another gate. The functionality of a gate is described by the so-called truth tables which map inputs to their outputs. Garbled circuits are Boolean circuits, where each gate's truth table is "garbled". Given two parties A and B, the four possible inputs of a garbled truth table are not bits, but random labels, e.g., encryption keys. Thus, one can envision a garbled truth table as a double-encrypted table, where one key corresponds to the input bit from party A and the other one from party B.
Next, we informally describe garbled circuits, for which a formalized description due to Bellare et al. [BHR12] is provided in Deliverable D5.4 "Final versions of tools for data sanitisation and computation". One party acts as the *garbler* and garbles the gates by creating the random labels. In other words, this party creates the garbled circuit. The other party acts as the *evaluator* and evaluates the garbled circuit. Note that the evaluator cannot learn labels for all possible inputs $0, 1$ per wire – as otherwise the evaluator can evaluate more than only its own input and learn more than only the agreed upon function output. Also, the garbler cannot learn the evaluator's input per wire – as this directly reveals the evaluator's sensitive input. To solve this problem, oblivious transfer is used. Thus, the garbler sends the evaluator only one of two possible labels l_0, l_1 (e.g., l_0 if the evaluator's input for the current gate is 0) and the garbler does not learn which one. Afterwards, the evaluator knows the garbled circuits with the garbled labels (which look random to the evaluator) and its own input labels and can evaluate (i.e., decrypt) each output label (i.e., next key) and use it to evaluate the next gate, etc. The garbler also produces an output translation table, which maps the final random output label to its corresponding plaintext result.

Converting between secret sharing and garbling

To use both schemes – secret sharing and garbled circuits – in one protocol, conversions between their value representations – secret shares and garbled values – are required.
To convert an additive secret share to a garbled value, one executes a garbled addition circuit where the secret shares are the inputs. The other direction, garbling to secret sharing, requires a garbled subtraction circuit C. Let x be the value to convert. Then, the garbler, e.g., Party A, samples a random value r, and circuit C receives inputs $[\![x]\!], [\![r]\!]$ and outputs $[\![x-r]\!]$ (using modulo) to the other party, who is allowed to decode this garbling and learn $x-r$. Now, A holds r and B holds $x-r$, which is a secret sharing of x. For further details and more efficient conversions we refer to [DSZ15].

3.3 Solutions

In this section, we present the different tools to enable UC3.
In Section 3.3.1, we detail DPtool which provides a REST API as well as a GUI for differential privacy mechanisms to support the sanitization of sensitive data. DPtool covers the local model of differential privacy, i.e., sanitization is applied during ingestion before storage – option 1 from Section 3.1. Also, DPtool supports the central model, i.e., sanitization is applied on centrally stored data for selected analysis functions – option 2 from Section 3.1.
In Section 3.3.2, we describe MIA, which helps data scientists to parameterize differential privacy in the context of machine learning and provides a membership inference analysis. MIA covers the central model, however, it can also cover the local model.
In Section 3.3.3, we present DPsc which securely computes the differentially private median. DPsc covers the hybrid model – option 3 from Section 3.1 – i.e., it closes the gap between the local and central model.

3.3.1 DPtool: Toolbox for Differential Privacy Mechanisms

DPtool permits to apply various anonymization methods on an input data set stored as comma-separated values (CSV file) and produces an anonymized output data set.

REST API

The DPtool exposes its services via a REST-API (REpresentational State Transfer). The REST service is implemented with Spring. Spring is a Java based framework for the development of web applications and supports the implementation of restful services [Spr0Z]. The anonymization methods are implemented in Java as well.
The paths for the input and output file as well as the parameters for the anonymization methods are provided as parameters to the REST API in JSON format. For each anonymization method, the column name, on which the anonymization will be applied, is required as well as the sensitivity and privacy parameter ε. The Gaussian mechanism requires an additional parameter δ, which is typically chosen to be negligible in the data size [DR14]. The sensitivity is the largest impact any individual (added/removed data point) can have on a function evaluation (see Definitions 3, 5).
The main anonymization methods exposed by the REST API are listed in Table 3.1 where input/output path are omitted for readability.

Method	Parameters
Laplace mechanism adds noise from a Laplace distribution; parameterized with sensitivity Δf and privacy parameter ε as in Definition 3.	`{ "column": <string>, "epsilon": <float>, "sensitivity": <float> }`
Geometric mechanism adds noise from a symmetric geometric distribution to integer counts; parameterized with privacy parameter ε as in Definition 4.	`{ "column": <string>, "epsilon": <float> }`
Gauss mechanism adds noise from a Gauss distribution; parameterized with sensitivity $\Delta_2 f$ and privacy parameters ε, δ as in Definition 5.	`{ "column": <string>, "epsilon": <float>, "sensitivity": <float>, "delta": <float> }`
Exponential mechanism probabilistically selects an output based on its utility score; parameterized with privacy parameter ε as in Definition 6. Utility scoring is provided for two applications: *i)* median (based on ranks, see also Section 3.3.3) and *ii)* similar words (based on Levenshtein distance).	`{ "column": <string>, "epsilon": <float> }`
Geo-Indistinguishability mechanism randomizes a location, expressed as latitude/-longitude, via privacy parameter ε to satisfy Definition 7.	`{ "latitudeColumn": <string>, "longitudeColumn": <string>, "epsilon": <float> }`

Table 3.1: DPtool: Main anonymization methods

Further methods are provided based on these main mechanism, as detailed in the following.

DP average uses the Laplace mechanism to compute a differentially private average with bounded results (parameters `min`, `max`) as described in Li et al. [LLSY16, Algorithm 2.3].

DP sum calls the Laplace mechanism without a sensitivity parameter and automatically detects the required sensitivity for the sum function based on the provided data (assuming that the data contains $\min \mathcal{U}, \max \mathcal{U}$).

Latitude/Longitude applies the geo-indistinguishability mechanism where only one of the coordinates (either latitude or longitude) are provided.

IP-geo randomizer maps IP addresses to coordinates before calling the geo-indistinguishability mechanism based on mapping data, which can be specified in the application settings.

DPtool also supports some anonymization techniques not based on perturbation and differential privacy:

Removal sets all values in the specified column to the empty string; thus, the structure of the data (e.g., columns) remains to avoid parsing errors in case of post-processing.

Removal by delimiter removes values before/after (defined via parameter `area`) pre-defined delimiters, i.e., "." (IPv4), ":" (MAC/IPv6), "@" (email).

GUI

SAP User Interface for HTML 5 (SAPUI5) is the basis for the graphical user interface. SAPUI5 is a framework to develop responsive web applications supporting HTML5 standards [SAP20], which provides convenient JavaScript libraries and is suited for model-view-controller (MVC) architectures. The model component details the data-related logic and is expressed in JavaScript. The view component describes the user interface with HTML/XML. The controller component connects the model and view components and processes incoming requests, e.g., actions taken via the view (do apply anonymization) will update the model (applied anonymization). Configurations, e.g., button labels (for localization), are stored in i18n property files.
Figures 3.2–3.4 visualize the steps in DPtool during the anonymization process.

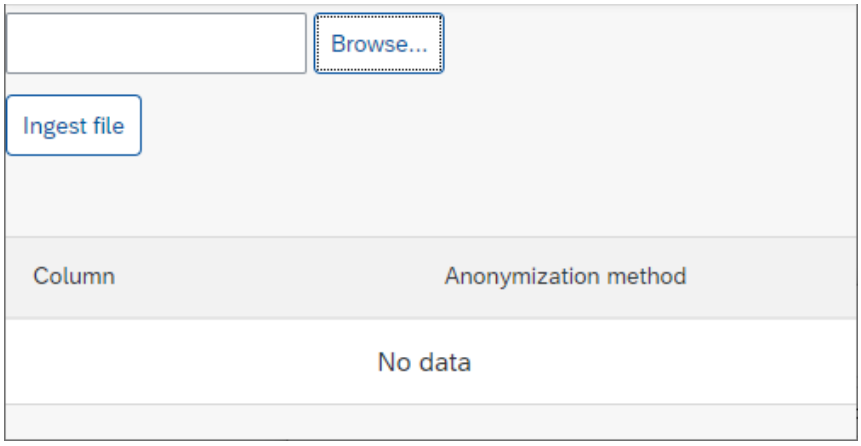


Figure 3.2: DPtool: Step 1 – data selection and ingestion

Initially, a client opens a browser and opens the URL corresponding to the running DPtool instance. First, the data is selected by browsing for it via a file dialog opened by pressing "Browse" and ingested by pressing "Ingest file". This initial selection step is shown in Figure 3.2. Then, the anonymization methods per named column can be selected via drop-down menu as in Figure 3.3. After selecting a desired method, its required parameters and input fields appear as in Figure 3.4, such that the methods can be parameterized accordingly. Finally, by pressing "anonymize", the methods are applied on the data and the data can be downloaded with a report detailing the selected anonymization mechanisms and parameterization.
Furthermore, during the parameterization step (Figure 3.4), the tool validates the parameters and provides tooltips for the mechanism selection, parameter selection, and highlights incorrect parameters with a red border as seen in Figure 3.5.

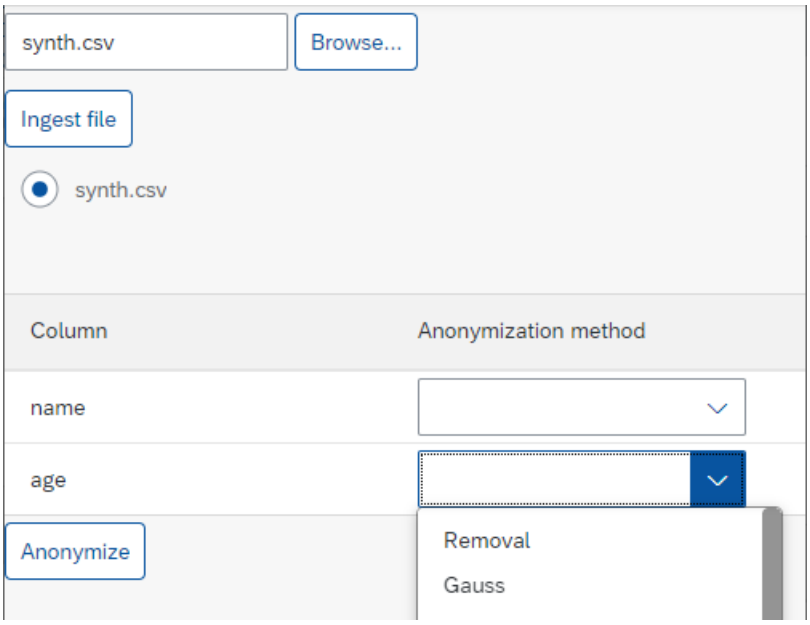


Figure 3.3: DPtool: Step 2 – method selection

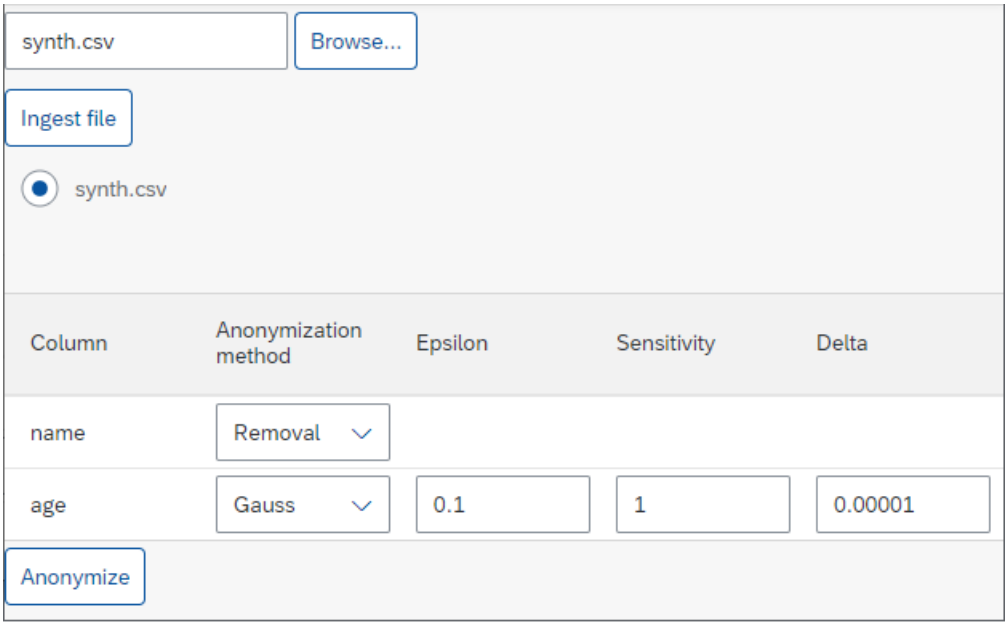


Figure 3.4: DPtool: Step 3 – method parameterization

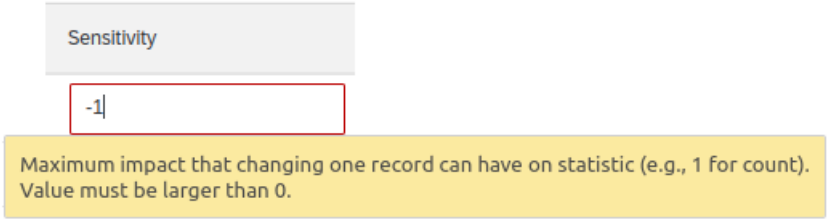


Figure 3.5: DPtool: Parameter validation and tooltip

3.3.2 MIA: Analysis for Membership Inference Attack

The MIA tool is a service to simulate membership inference attacks (MIA) on machine learning models. It can be used to assure ML quality with regards to accuracy of the anonymized models as well as quantify the privacy by the success probability for membership inference attacks.

Architecture

Theoretical details for membership inference are given in Deliverable D5.2 "First Report on Privacy Metrics and Data Sanitisation" and the architecture of the MIA tool is detailed in Deliverable D5.1 "First Version of Data Sanitisation Tools" and we briefly recall the architecture here. MIA is mainly realized with Python and the GUI is written in SAPUI5 like DPtool. The MIA REST service runs in a Tornada HTTP Server. The configurations, ML models and its parameters are stored within a MongoDB data base system. Data sets, uploaded to MIA, are stored as files.

Usage

Figures 3.6–3.8 are screenshots of the step-by-step guide to initialize a report with MIA. Initially, a client opens a browser and opens the URL corresponding to the running MIA instance.
Step 1, i.e., Figure 3.6, shows the selection of a named machine learning model for which an analysis is to be created. Step 2, i.e., Figure 3.7, shows the selection of the training data for the previously selected model. Step 3, i.e., Figure 3.8, shows the parameterization of the model. This simple setup suffices to start the training process. Afterwards, an analysis of the membership inference risk and the utility with and without differential privacy can be compared.

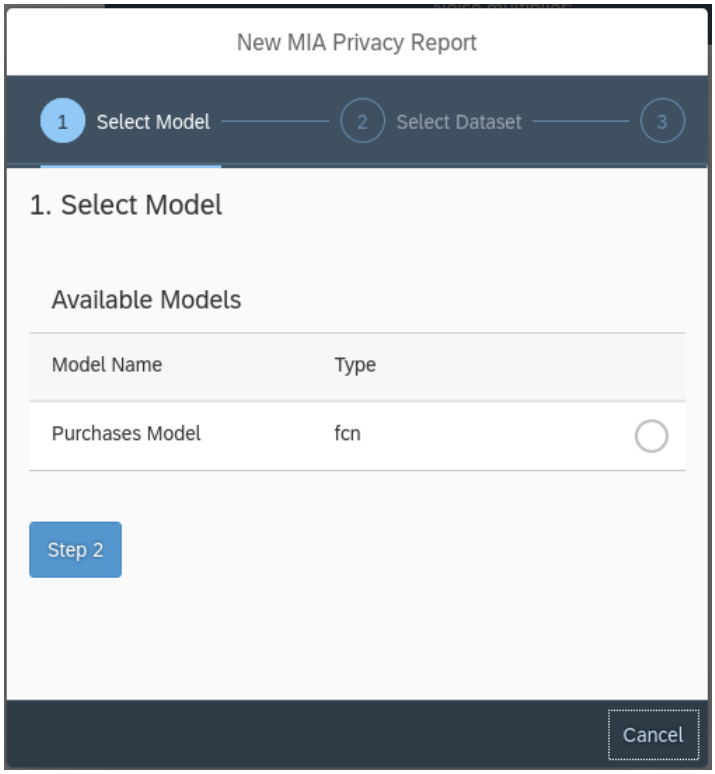


Figure 3.6: MIA: Step 1 – model selection

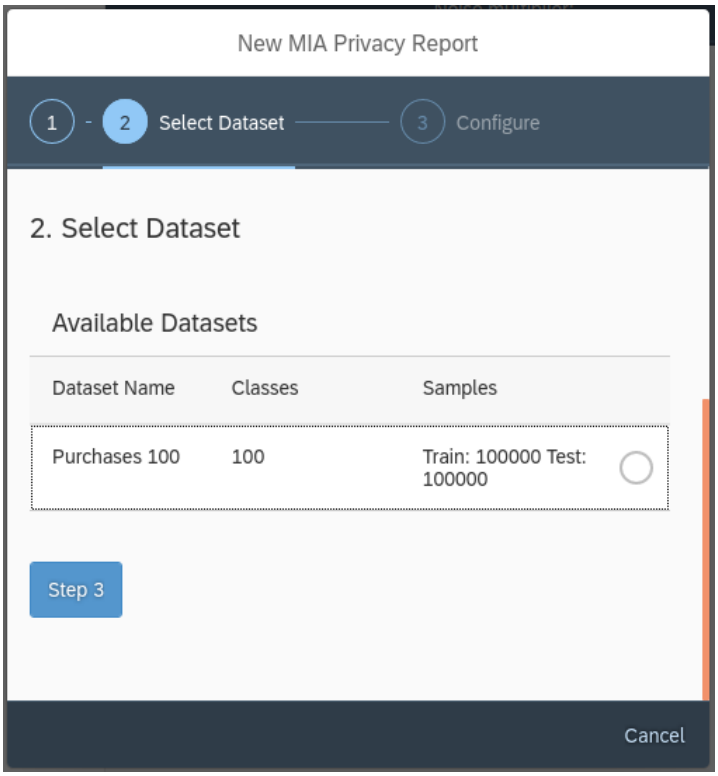


Figure 3.7: MIA: Step 2 – data selection

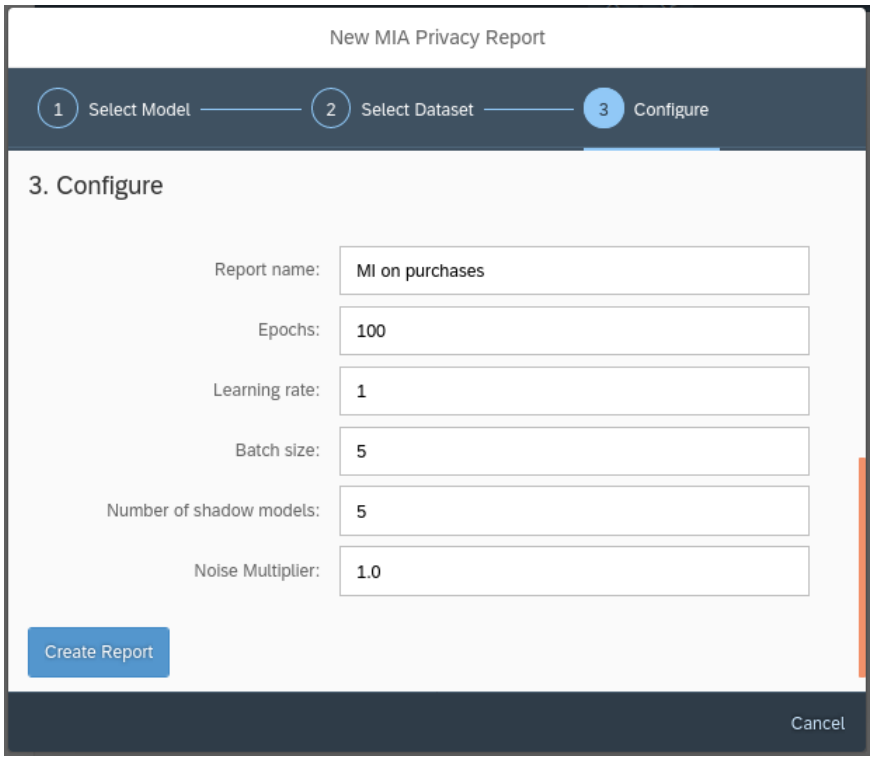


Figure 3.8: MIA: Step 3 – parameterization

3.3.3 DPsc: Secure Computation of Differentially Private Statistics

The theoretical insights that led to DPsc are detailed in Deliverable D5.4 "Final versions of tools for data sanitisation and computation" based on a published paper [BK20b] and an extension for the multi-party setting was detailed in Deliverable D5.5 "Report on data sanitisation and computation" and also published as a paper [BK20a]. In this section, we focus on the tool itself, its underlying technology and usage.

Rank-based Statistics

DPsc is a protocol to securely compute differentially private rank-based statistics over distributed data held by two parties. Order statistics, also called ranked-based statistics, are defined over a sorted data set D. The *rank* is the first, zero-based position of an element in a sorted data set. Figure 3.9 visualizes the ranks for a sorted data set and includes examples for rank-based statistics, namely, the minimum, median, and maximum of D. While DPsc supports any rank-based statistics by, e.g., padding the data accordingly [AMP10], we focus on the median, which is the element that roughly splits the sorted data in half.

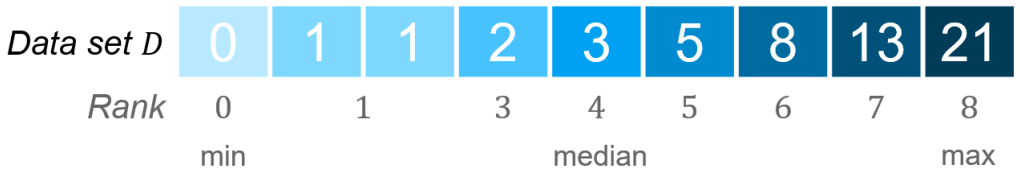


Figure 3.9: Sorted data set D with ranks per unique datum

Overview

Next, we provide an overview of DPsc which is described in detail in deliverable D5.4 "Final versions of tools for data sanitisation and computation".
Firstly, if the data size n is not sublinear in the size of the domain $|\mathcal{U}|$, i.e., $n > \log_2 |\mathcal{U}|$, then DPsc first prunes the data according to Aggarwal et al. [AMP10]: The parties A and B compute their local median values m_A and m_B, respectively, over their own data and securely compare it via garbled circuits to learn $c = m_A < m_B$. If the resulting bit c is 1, i.e., m_A is smaller than m_B, then A discards the upper half of its sorted data and B discards its lower data half. If c is 0, they do the opposite. This is repeated until the data is small enough, i.e., sublinear in $|\mathcal{U}|$.
Secondly, the parties securely merge their pre-sorted (potentially pruned) data via garbled circuits and learn a secret shared version of the merged, sorted data.
Thirdly, the parties compute the selection probabilities over the secret shared data. This can be done via multiplication with known values and subtraction of secret shared values.
Finally, the parties sample the DP median via inverse transform sampling over the previously computed selection probabilities.

Architecture

DPsc is implemented with the mixed-protocol secure computation framework ABY [DSZ15] which is actively maintained. ABY provides basic secure protocols for, e.g., addition and comparison. It supports garbled circuits as well as secret sharing and provides efficient conversions between these schemes.

ABY programs are written in C++ and define a circuit `circ`, i.e., all operations are expressed as logical gates. For example, function `circ->PutGTGate(inputa, inputb)` adds a comparison gate to the circuit, which evaluates to 1 if `inputa` is greater than `inputb`.
For development purposes, ABY can be setup as described on ABY's github page[2]. Briefly, on a Linux system one installs the requirements[3], clones the repository, and builds ABY, including example applications, as follows:

```
cd ABY/ && mkdir build && cd build
cmake ..  -DABY_BUILD_EXE=On
make
```

The source structure of an ABY program, like DPsc, is as follows. The program folder contains

- file `CMakeLists.txt` to build the project with cmake
- file `dp_sc.cpp` which contains the main function and the logic to parse the parameters
- folder `common` containing
  - file `dp_sc.cpp` containing the main logic
  - file `dp_sc.h` containing the interface definition

To create an executable from a program, one calls `cmake . && make .` in the program folder. After creating an executable, the executable can be transferred to the participating parties. Then, the parties have to execute it as explained for DPsc next.

Usage

DPsc is a command line tool which acts as server or client depending on how it is invoked. To jointly execute DPsc, one party invokes it as a server the other one as a client. The only result is the differentially private median over their joint data set – nothing else is revealed about the data. DPsc has the following mandatory parameters:

```
-r [Role: 0/1 for server/client, required]
-f [Data set file (one value per line), required]
-n [Number of elements, required]
-b [Bit-length, required]
```

DPsc supports the following optional parameters:

```
-e [Epsilon, default: ln(2), optional]
-i [IP-address, default: localhost, optional]
-p [Port, default: 7766, optional]
-c [Convert to arithmetic sharing for ADD/MUL], optional]
-m [Min universe element (default: min of input), optional]
-M [Max universe element (default: max of input), optional]
-N [Number of nonces (per party) for unbiased sampling, optional]
-B [Use biased sampling (uses only one nonce, i.e., -N 1), optional]
```

[2] `https://github.com/encryptogroup/ABY`
[3] `g++, make, cmake, libgmp-dev, libssl-dev, libboost-all-dev.`

```
-s [Symmetric Security Bits, default: 128, optional]
-a [Accuracy, default: 0.90, optional]
-o [Print debug output, optional]
```

As an example, consider the first party executing

```
./dp_sc -r 0 -f data_1.text -n 10000 -e 0.1 -b 64 -N 30
```

and the second party executing

```
./dp_sc -r 1 -f data_2.text -n 10000 -e 0.1 -b 64 -N 30
```

The first starts DPsc as a server, reads the first 10,000 values from data set `data_1.txt`, sets ε to 0.1, defines bit-length 64, and uses 30 input nonces for rejection sampling. The second does the same, however with the role of client and with another data set. In a WAN setting, the IP address of the server must be specified as well with parameter `-i` and, potentially, the port with parameter `-p`. Next, we further detail some parameters.

Parameter `-c` adds conversions: While the main part of our protocol is written as a garbled circuit, we support converting arithmetic operations to secret sharing. In more detail, addition of secret values and multiplication with known values are costly in Boolean circuits (i.e., garbled circuits) but "free" in arithmetic circuits (i.e., secret sharing) in the sense that no interaction is required between the parties. Such conversion requires some interaction between the parties. However, in our evaluation (detailed in Deliverable D5.4 "Final versions of tools for data sanitisation and computation") a protocol execution with conversion is much faster in a wide-area network than one without conversion, as the conversion overhead is much smaller than the overhead for, e.g., multiple additions in Boolean circuits.

Note that the parties can input the minimum (`-m`) and maximum domain element (`-M`) if these are not contained in the data set. The minimum and maximum domain elements are required to satisfy differential privacy, as all neighboring data sets must be considered, which can contain any element from the entire data domain. Recall, a neighboring data set D' of data set D is just D with an element removed or added. We must compute non-zero selection probabilities for all possible output elements, i.e., domain elements in the case of the median, to satisfy differential privacy.

Parameter `-B` selects biased sampling, by default (unbiased) rejection sampling is used. Both methods refer to uniform sampling, i.e., outputting a random element from a fixed range where all elements are equally likely to be selected. Biased sampling is simpler but slightly favors smaller values, i.e., it is not perfectly uniformly random. These different sampling methods were implemented to evaluate if the former is more efficient than the latter. However, in our evaluation (detailed in Deliverable D5.4 "Final versions of tools for data sanitisation and computation") rejection sampling adds only negligible overhead while being unbiased. Biased sampling of integer range $[0,R)$ can be implemented for two semi-honest parties as follows: First, each party selects a random number, a so-called nonce; then, the parties compute x by XORing their nonces, and output $x \mod R$. However, R is secret as it is the normalization term (i.e., denominator) in Definition 6. Thus, R is computed on the sensitive data, which makes it sensitive as well, as it can leak information about the data.[4] The parties cannot learn R; however, an upper bound U can be derived by assuming the largest possible utility scores for each element. Thus, the parties input nonces from $[0,U]$. This sampling method is biased, as modulo R does not evenly divide the range

[4] For example, the normalization term can differ between neighboring data sets, which suffices to differentiate them and violate differential privacy.

of possible values for x, i.e., the modulo operation slightly favors smaller values as outputs. Rejection sampling, on the other hand, is unbiased. Here, we use multiple nonces (parameter `-N`). For each nonce per party, we combine them via XOR to x as before and compute r as the AND of x with a bit-mask b. The bits in b are zero until the first position where R has a one, and consists of only ones afterward (e.g., $b = 00111_2$ for $R = 00101_2$). Now, we reject r if it is larger than R and otherwise stop and output r. All nonces might be rejected, however, the probability is negligible in the number of nonces (i.e., 2^{-N} for N nonces). We use a fixed number of nonces, as secure computation does not allow conditional loops, e.g., executing a loop until a condition is met, when the condition is secret and the number of iteration steps might reveal something about the condition. For more details about the implemented sampling methods, we refer to deliverable D5.4 "Final versions of tools for data sanitisation and computation".
Parameter `-s` corresponds to the security parameter λ in Definition 2.
Parameter `-a` defines the selection accuracy, i.e., the probability to select an output from the remaining elements instead of the pruned ones (see Deliverable D5.4 "Final versions of tools for data sanitisation and computation" for details).
Parameter `-o` provides debug output during the computation. This parameter should only be used for demonstration purposes as it reveals all secret values and nonces!

3.4 Summary

UC3 considers a cloud-based data market, where different parties want to share insights over their data sets without revealing their data to each other. To allow such privacy-preserving analytics in a cloud-based environment, we presented different tools. Namely, DPtool, MIA, and DPsc, which enable the use case and cover the entire life-cycle of MOSAICrOWN, from ingestion and storage to analytics.
Next, we briefly summarize the different tools.

DPtool provides simple interfaces to apply differential privacy mechanisms on data sets to sanitize them during ingestion (i.e., only sanitized data is stored) or during analytics (i.e., data is stored in plaintext or encrypted). DPtool provides a programmatic interface in the form of a REST API allowing automatic sanitization. Furthermore, DPtool provides a graphical interface in the form of SAPUI5 which permits manual sanitization, e.g., to support analysts in evaluating different techniques and parameters with immediate feedback by displaying a subset of the sanitized data.

MIA supports data scientists in the parameterization of differential privacy in the context of machine learning, e.g., during ingestion or investigation of analytical functions of interest. MIA provides a graphical user interface based on SAPUI5 and assists data scientists and analysts in evaluating the protection guarantees of differential privacy via membership inference risks as well as the utility-privacy trade-offs inherent in any anonymization technique.

DPsc enables differentially private rank-based statistics over two distributed data sets (as envisioned in the use case) without sharing the data at all. In other words, the ingestion and storage phase are skipped and we go right to the analytical phase. DPsc combines secure computation and differential privacy, i.e., the parties collaboratively compute a DP mechanism (in this case the exponential mechanism) while only learning the mechanism's output but none of the inputs from the other party. DPsc is a command-line tool with few required

parameters, e.g., input data and privacy parameter, and multiple optional parameters (with sane defaults).

4. Conclusions

This deliverable reported on the prototypes developed during MOSAICrOWN to satisfy the use cases by the industry partners (EISI, MC, SAP SE). The prototypes leverage technologies developed in Work Packages 3–5.

Chapter 1 presented tools for Use Case 1. This use case, by EISI, considers data protection in the context of platforms for intelligent connected vehicles where, e.g., electrical vehicles and charging infrastructures can securely collect and exchange insights to improve planning and charger allocations. To realize this use case, different tools have been designed and developed. First, an automotive tool to facilitate data collection and ingestion from the electric vehicle to the data market. Second, a web tool to enable the authorized data access via a user interface. Third, a policy engine realizing access control, i.e., permitting or denying access, after parsing the corresponding MOSAICrOWN policy. Fourth, an encrypted filesystem providing confidentiality and enabling secure storage of the personal data.

Chapter 2 detailed tools for Use Case 2. This use case, by MC, focuses on financial institutions and their transaction-level data anonymization. The proof-of-concept tools provide a platform for different wrapping as well as data sanitization techniques to augment policy-based protection mechanisms which control data access, usage, and sharing. Moreover, the tools are flexible when considering different privacy regulations (e.g., GDPR and LGPD). The tools automatically identify the semantics of each field (in terms of type, data distribution, and content) and suggest for each field the wrapping techniques related to the selected privacy regulation. Also, visualization of data analytics and summary results are provided by the tools.

Chapter 3 described tools for Use Case 3. In this use case, by SAP SE, the goal is to enable privacy-preserving analytics for cloud-based data markets. In more detail, businesses want to share insights over their business-sensitive operational data and personal customer experience data, without revealing the underlying sensitive data. To realize this use case, three tools where designed to cover different aspects of privacy-preserving analytics: DPtool, MIA, and DPsc. DPtool provides various mechanisms for anonymization with differentially privacy, such as additive noise sampled from the Laplace distribution (for reals) or the Geometric distribution (for integers). MIA aims to help data scientists during the parameterization of differential privacy mechanisms in the context of machine learning and provides a risk-based evaluation of the applied parameterization. DPsc combines differential privacy with cryptographic techniques to enable distributed anonymization on data of multiple parties releasing only the anonymized results for rank-based statistics which includes percentiles such as the median (the 50-th percentile).

Bibliography

[ABCP13] M. E. Andrés, N. E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, November 2013.

[AMP10] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the median (and other elements of specified ranks). *Journal of Cryptology*, 23:373–401, 2010.

[Bea91] D. Beaver. Efficient multiparty protocols using circuit randomization. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, USA, August 1991.

[Bea96] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proc. of the Annual ACM Symposium on Theory of Computing (STOC)*, Philadelphia, Pennsylvania, May 1996.

[BHR12] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, Raleigh, NC, USA, October 2012.

[BK20a] J. Boehler and F. Kerschbaum. Secure multi-party computation of differentially private median. In *Proc. of the 29th USENIX Security Symposium*, Virtual event, August 2020.

[BK20b] J. Boehler and F. Kerschbaum. Secure sublinear time differentially private median computation. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2020.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. of the International Workshop on Managing Requirements Knowledge (MARK)*, June 1979.

[CM16] R. Coppola and M. Morisio. Connected car: technologies, issues, future trends. *ACM Computing Surveys (CSUR)*, 49(3):1–36, 2016.

[CO15] T. Chou and C. Orlandi. The simplest protocol for oblivious transfer. In *Proc. of thhe International Conference on Cryptology and Information Security in Latin America (LATINCRYPT)*, Guadalajara, Mexico, August 2015.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6), November 1976.

[DR14] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[DSZ15] D. Demmler, T. Schneider, and M. Zohner. Aby - a framework for efficient mixed-protocol secure two-party computation. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2015.

[Fou] OpenJS Fountation. Openjs fountation. `https://nodejs.org/en/`. (Accessed on 06/01/2021).

[GRS12] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing*, 2012.

[HMFS17] X. He, A. Machanavajjhala, C. Flynn, and D. Srivastava. Composing differential privacy and secure computation: A case study on scaling private record linkage. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)*, Dallas, TX, USA, October-November 2017.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious transfers efficiently. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, USA, August 2003.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of the Annual ACM Symposium on Theory of Computing (STOC)*, Chicago, IL, USA, May 1988.

[Koo21] J. Kook. The design, implementation, demonstration of the architecture, service framework, and applications for a connected car. *KSII Transactions on Internet & Information Systems*, 15(2), 2021.

[LLSY16] N. Li, M. Lyu, D. Su, and W. Yang. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, & Trust. Morgan & Claypool, 2016.

[MG20] A. M. Medina and J. Gillenwater. Duff: A dataset-distance-based utility function family for the exponential mechanism. *arXiv preprint arXiv:2010.04235*, 2020. `https://arxiv.org/abs/2010.04235`.

[MPRV09] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. In *Proc. of the Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, USA, August 2009.

[MT07] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proc. of the 48th Annual Symposium on Foundations of Computer Science (FOCS)*, Providence, RI, USA, October 2007.

[Pol] Polestar. Polestar 2 google android automotive os | polestar us. `https://www.polestar.com/us/polestar-2/google-polestar/`. (Accessed on 05/25/2021).

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. *Technical Memo TR-81*, 1981.

[SAP20] SAP. What is SAPUI5?, 2020. `https://searchsap.techtarget.com/definition/SAPUI5`.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), November 1979.

[sit] sitepoint. A beginner's guide to pug. `https://www.sitepoint.com/a-beginners-guide-to-pug/`. (Accessed on 06/01/2021).

[Spr0Z] Spring. Introduction to spring framework, 2020-09-03T10:40:46.000Z.

[Tec] HERE Technologies. Here maps: Developer guide - get started. `https://developer.here.com/documentation/maps/3.1.25.0/dev_guide/topics/get-started.html`. (Accessed on 06/01/2021).

[Yao86] A.C.-C. Yao. How to generate and exchange secrets. In *Proc. of the 27th Annual Symposium on Foundations of Computer Science (FOCS)*, Toronto, Canada, October 1986.